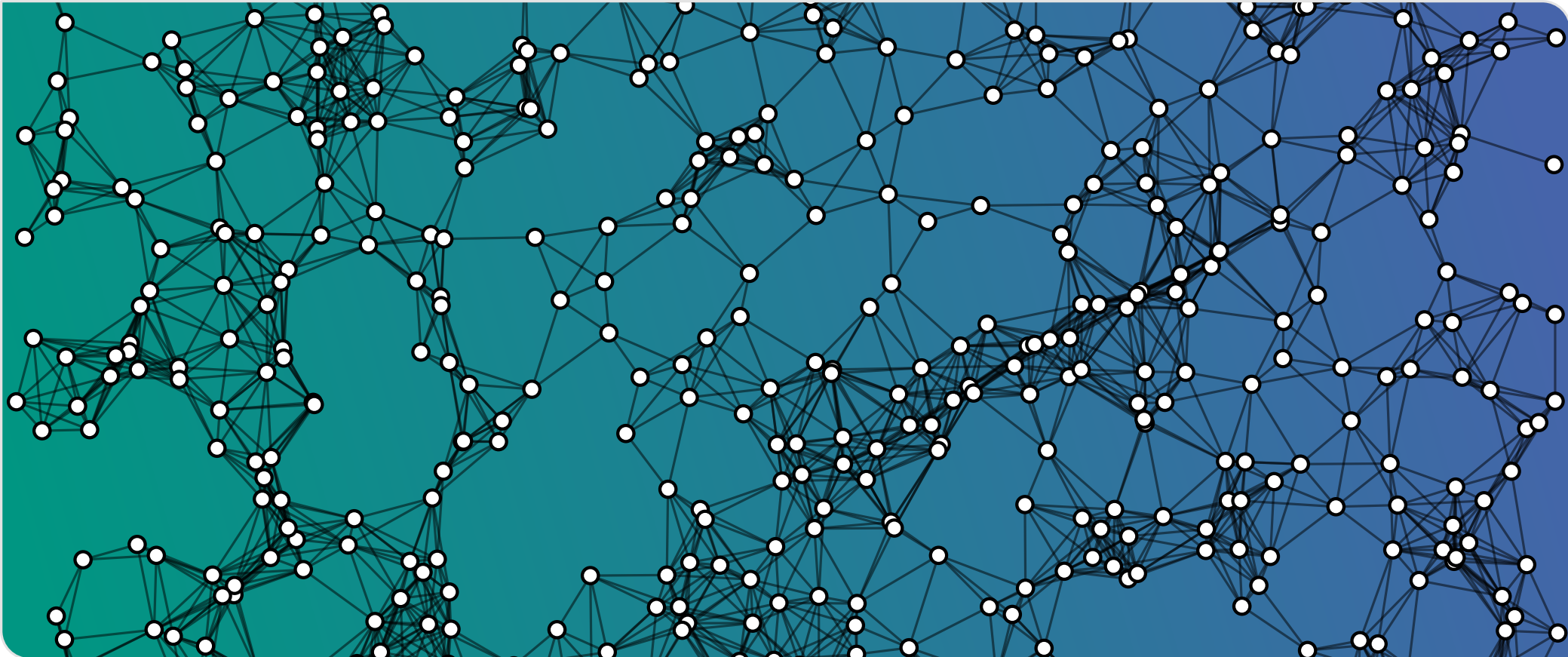


Parametrisierte Algorithmen

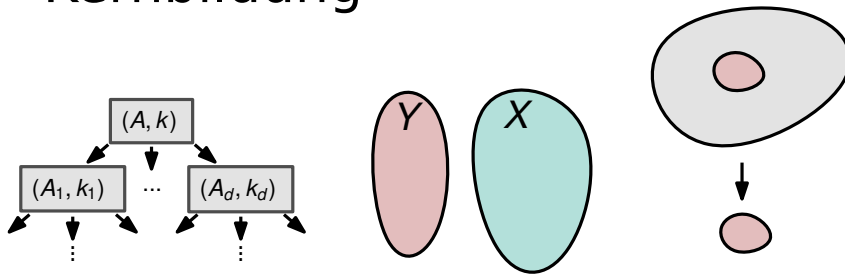
Untere Schranken: Reduktionen und W-Hierarchie



Inhalt

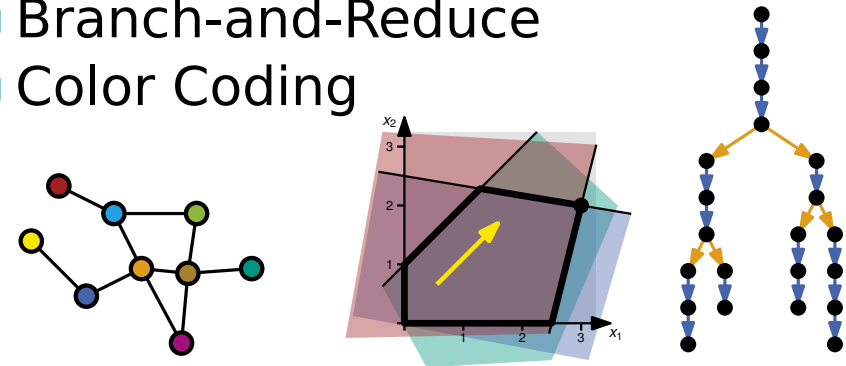
Basic Toolbox

- beschränkte Suchbäume
- iterative Kompression
- Kernbildung



Erweiterte Toolbox

- lineare Programme
- Branch-and-Reduce
- Color Coding



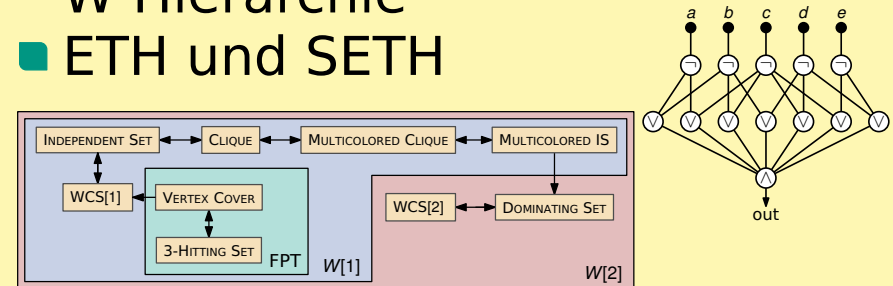
Baumweite

- dynamische Programme
- chordale & planare Graphen
- Courcelles Theorem

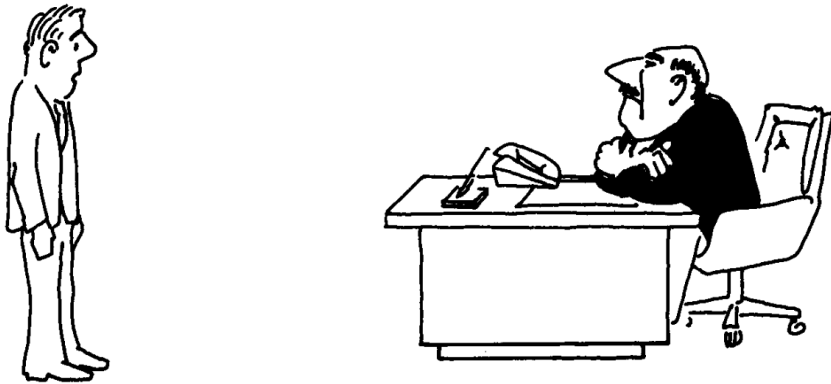


Untere Schranken

- parametrisierte Reduktionen
- boolesche Schaltkreise und die W-Hierarchie
- ETH und SETH



Was bringen untere Schranken?

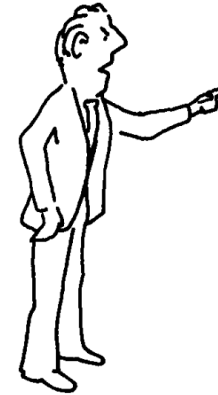


“I can’t find an efficient algorithm, I guess I’m just too dumb.”

Was bringen untere Schranken?



“I can’t find an efficient algorithm, I guess I’m just too dumb.”

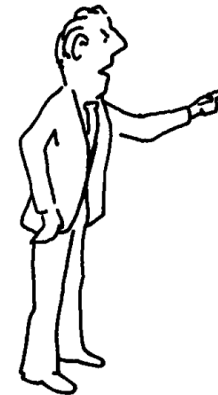


“I can’t find an efficient algorithm, because no such algorithm is possible!”

Was bringen untere Schranken?



“I can’t find an efficient algorithm, I guess I’m just too dumb.”

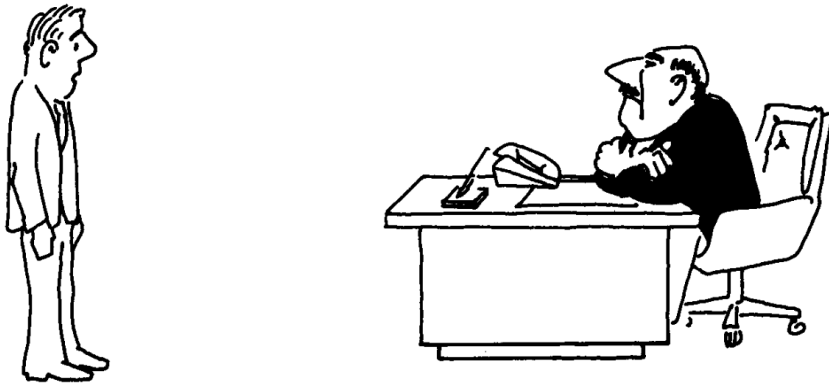


“I can’t find an efficient algorithm, because no such algorithm is possible!”

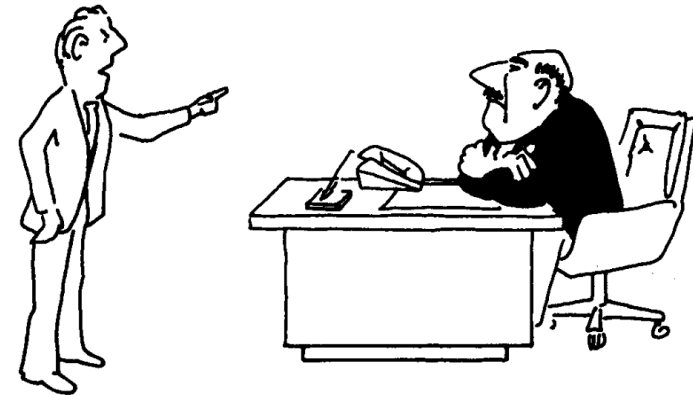
Problem

- nicht-Existenz eines Algorithmus ist schwer zu zeigen

Was bringen untere Schranken?



“I can’t find an efficient algorithm, I guess I’m just too dumb.”



“I can’t find an efficient algorithm, because no such algorithm is possible!”

Problem

- nicht-Existenz eines Algorithmus ist schwer zu zeigen

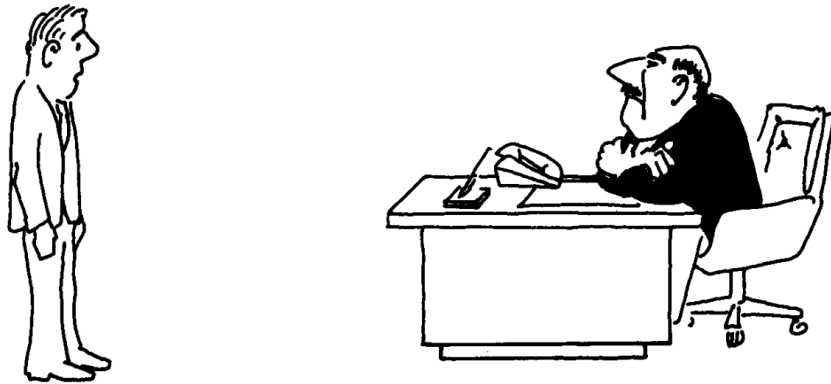
Lösung

- zeige: mein Problem effizient lösbar
 \Rightarrow ein bekanntes schwieriges Problem effizient lösbar

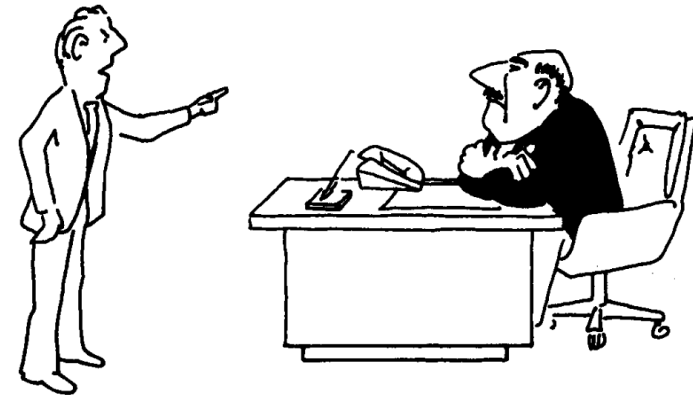


“I can’t find an efficient algorithm, but neither can all these famous people.”

Was bringen untere Schranken?



“I can’t find an efficient algorithm, I guess I’m just too dumb.”



“I can’t find an efficient algorithm, because no such algorithm is possible!”

Problem

- nicht-Existenz eines Algorithmus ist schwer zu zeigen

Lösung

- zeige: mein Problem effizient lösbar \Rightarrow ein bekanntes schwieriges Problem effizient lösbar
- Werkzeug: Reduktionen zwischen Problemen



“I can’t find an efficient algorithm, but neither can all these famous people.”

Polynomielle Reduktionen

Reduktion von Problem \mathcal{L} zu Problem \mathcal{L}'

- bildet jede Instanz I von \mathcal{L} auf eine Instanz I' von \mathcal{L}' ab
- I ist ja-Instanz $\Leftrightarrow I'$ ist ja-Instanz

Polynomielle Reduktionen

Reduktion von Problem \mathcal{L} zu Problem \mathcal{L}'

- bildet jede Instanz I von \mathcal{L} auf eine Instanz I' von \mathcal{L}' ab
- I ist ja-Instanz $\Leftrightarrow I'$ ist ja-Instanz
- die Abbildung muss in polynomieller Zeit berechnet werden können

Polynomielle Reduktionen

Reduktion von Problem \mathcal{L} zu Problem \mathcal{L}'

- bildet jede Instanz I von \mathcal{L} auf eine Instanz I' von \mathcal{L}' ab
- I ist ja-Instanz $\Leftrightarrow I'$ ist ja-Instanz
- die Abbildung muss in polynomieller Zeit berechnet werden können

Folgerung

- polynomieller Algorithmus für $\mathcal{L}' \Rightarrow$ polynomiellen Algorithmus für \mathcal{L}

Polynomielle Reduktionen

Reduktion von Problem \mathcal{L} zu Problem \mathcal{L}'

- bildet jede Instanz I von \mathcal{L} auf eine Instanz I' von \mathcal{L}' ab
- I ist ja-Instanz $\Leftrightarrow I'$ ist ja-Instanz
- die Abbildung muss in polynomieller Zeit berechnet werden können

Folgerung

- polynomieller Algorithmus für $\mathcal{L}' \Rightarrow$ polynomiellen Algorithmus für \mathcal{L}
- mein Problem \mathcal{L}' effizient lösbar \Rightarrow bekanntes schwieriges Problem \mathcal{L} effizient lösbar
(zumindest, wenn „effizient“ = „polynomiell“)

Polynomielle Reduktionen

Reduktion von Problem \mathcal{L} zu Problem \mathcal{L}'

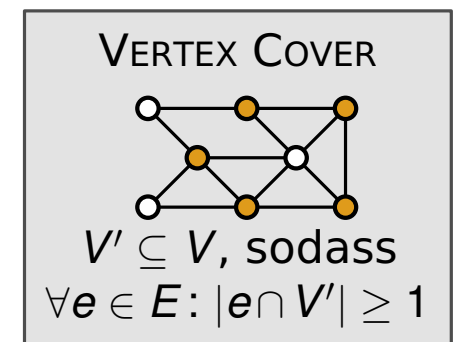
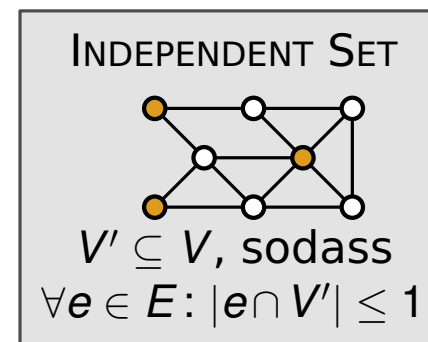
- bildet jede Instanz I von \mathcal{L} auf eine Instanz I' von \mathcal{L}' ab
- I ist ja-Instanz $\Leftrightarrow I'$ ist ja-Instanz
- die Abbildung muss in polynomieller Zeit berechnet werden können

Folgerung

- polynomieller Algorithmus für $\mathcal{L}' \Rightarrow$ polynomiellen Algorithmus für \mathcal{L}
- mein Problem \mathcal{L}' effizient lösbar \Rightarrow bekanntes schwieriges Problem \mathcal{L} effizient lösbar
(zumindest, wenn „effizient“ = „polynomiell“)

Beispiel

- reduziere INDEPENDENT SET auf VERTEX COVER



Polynomielle Reduktionen

Reduktion von Problem \mathcal{L} zu Problem \mathcal{L}'

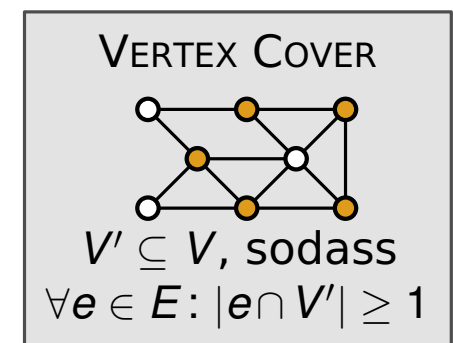
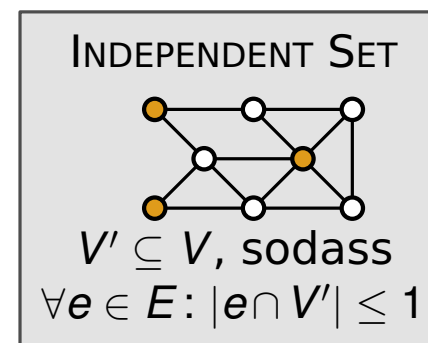
- bildet jede Instanz I von \mathcal{L} auf eine Instanz I' von \mathcal{L}' ab
- I ist ja-Instanz $\Leftrightarrow I'$ ist ja-Instanz
- die Abbildung muss in polynomieller Zeit berechnet werden können

Folgerung

- polynomieller Algorithmus für $\mathcal{L}' \Rightarrow$ polynomiellen Algorithmus für \mathcal{L}
- mein Problem \mathcal{L}' effizient lösbar \Rightarrow bekanntes schwieriges Problem \mathcal{L} effizient lösbar
(zumindest, wenn „effizient“ = „polynomiell“)

Beispiel

- reduziere INDEPENDENT SET auf VERTEX COVER
- triviale Reduktion: G hat IS der Größe $k \Leftrightarrow G$ hat VC der Größe $n - k$



Polynomielle Reduktionen

Reduktion von Problem \mathcal{L} zu Problem \mathcal{L}'

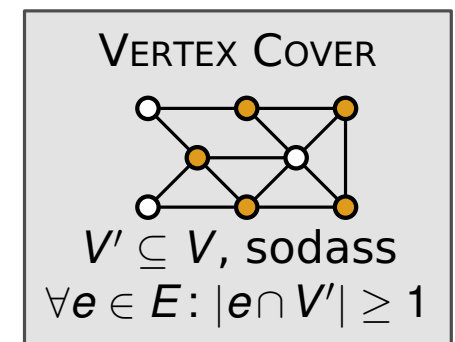
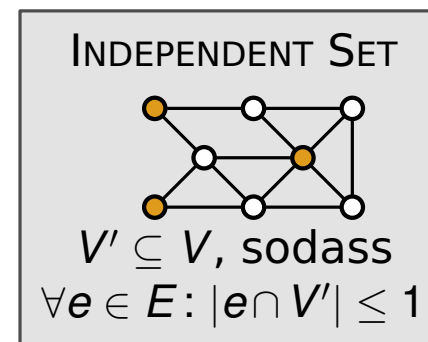
- bildet jede Instanz I von \mathcal{L} auf eine Instanz I' von \mathcal{L}' ab
- I ist ja-Instanz $\Leftrightarrow I'$ ist ja-Instanz
- die Abbildung muss in polynomieller Zeit berechnet werden können

Folgerung

- polynomieller Algorithmus für $\mathcal{L}' \Rightarrow$ polynomiellen Algorithmus für \mathcal{L}
- mein Problem \mathcal{L}' effizient lösbar \Rightarrow bekanntes schwieriges Problem \mathcal{L} effizient lösbar
(zumindest, wenn „effizient“ = „polynomiell“)

Beispiel

- reduziere INDEPENDENT SET auf VERTEX COVER
- triviale Reduktion: G hat IS der Größe $k \Leftrightarrow G$ hat VC der Größe $n - k$
- also: $VC \in P \Rightarrow IS \in P$



Polynomielle Reduktionen

Reduktion von Problem \mathcal{L} zu Problem \mathcal{L}'

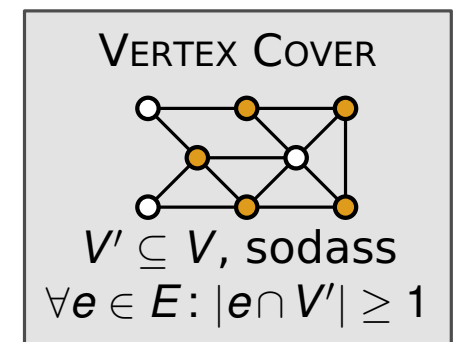
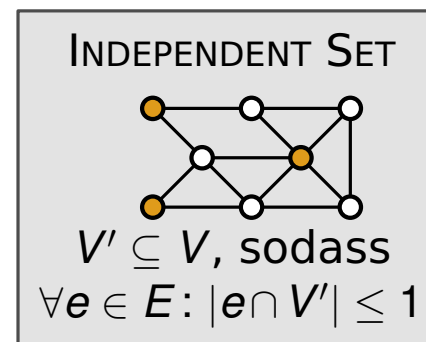
- bildet jede Instanz I von \mathcal{L} auf eine Instanz I' von \mathcal{L}' ab
- I ist ja-Instanz $\Leftrightarrow I'$ ist ja-Instanz
- die Abbildung muss in polynomieller Zeit berechnet werden können

Folgerung

- polynomieller Algorithmus für $\mathcal{L}' \Rightarrow$ polynomiellen Algorithmus für \mathcal{L}
- mein Problem \mathcal{L}' effizient lösbar \Rightarrow bekanntes schwieriges Problem \mathcal{L} effizient lösbar
(zumindest, wenn „effizient“ = „polynomiell“)

Beispiel

- reduziere INDEPENDENT SET auf VERTEX COVER
- triviale Reduktion: G hat IS der Größe $k \Leftrightarrow G$ hat VC der Größe $n - k$
- also: $VC \in P \Rightarrow IS \in P$
- gilt auch $VC \in FPT \Rightarrow IS \in FPT$?



Polynomielle Reduktionen

Reduktion von Problem \mathcal{L} zu Problem \mathcal{L}'

- bildet jede Instanz I von \mathcal{L} auf eine Instanz I' von \mathcal{L}' ab
- I ist ja-Instanz $\Leftrightarrow I'$ ist ja-Instanz
- die Abbildung muss in polynomieller Zeit berechnet werden können

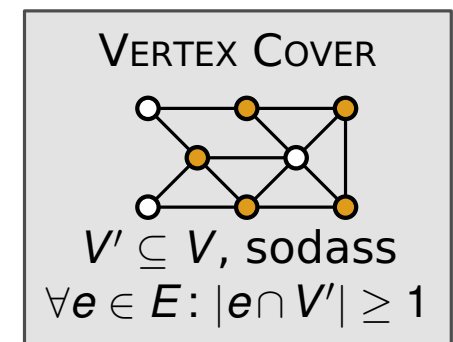
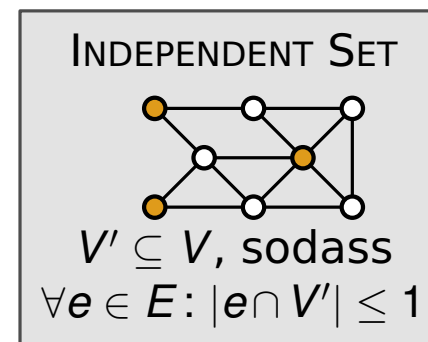
Folgerung

- polynomieller Algorithmus für $\mathcal{L}' \Rightarrow$ polynomiellen Algorithmus für \mathcal{L}
- mein Problem \mathcal{L}' effizient lösbar \Rightarrow bekanntes schwieriges Problem \mathcal{L} effizient lösbar
(zumindest, wenn „effizient“ = „polynomiell“)

Beispiel

- reduziere INDEPENDENT SET auf VERTEX COVER
- triviale Reduktion: G hat IS der Größe $k \Leftrightarrow G$ hat VC der Größe $n - k$
- also: $VC \in P \Rightarrow IS \in P$
- gilt auch $VC \in FPT \Rightarrow IS \in FPT$?
→ nein, der Parameter wird zu groß

(implizite Annahme: Lösungsgröße ist Parameter)



Polynomielle Reduktionen

Reduktion von Problem \mathcal{L} zu Problem \mathcal{L}'

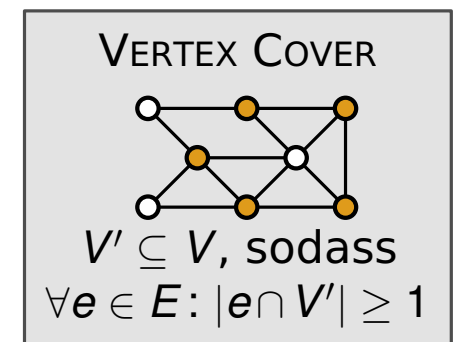
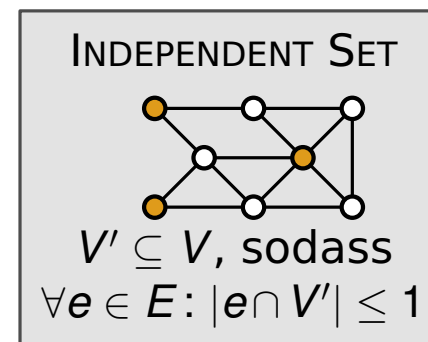
- bildet jede Instanz I von \mathcal{L} auf eine Instanz I' von \mathcal{L}' ab
- I ist ja-Instanz $\Leftrightarrow I'$ ist ja-Instanz
- die Abbildung muss in polynomieller Zeit berechnet werden können

Folgerung

- polynomieller Algorithmus für $\mathcal{L}' \Rightarrow$ polynomiellen Algorithmus für \mathcal{L}
- mein Problem \mathcal{L}' effizient lösbar \Rightarrow bekanntes schwieriges Problem \mathcal{L} effizient lösbar
(zumindest, wenn „effizient“ = „polynomiell“)

Beispiel

- reduziere INDEPENDENT SET auf VERTEX COVER
- triviale Reduktion: G hat IS der Größe $k \Leftrightarrow G$ hat VC der Größe $n - k$
- also: $VC \in P \Rightarrow IS \in P$
- gilt auch $VC \in FPT \Rightarrow IS \in FPT$?
→ nein, der Parameter wird zu groß
- für „effizient“ = „FPT“ benötigt man andere Reduktionen
(implizite Annahme: Lösungsgröße ist Parameter)



Parametrisierte Reduktionen

Reduktion von Problem \mathcal{L} zu Problem \mathcal{L}'

- bildet jede Instanz (I, k) von \mathcal{L} auf eine Instanz (I', k') von \mathcal{L}' ab
- (I, k) ist ja-Instanz $\Leftrightarrow (I', k')$ ist ja-Instanz; mit $k' \leq g(k)$
- die Abbildung muss in FPT-Zeit $(f(k) \cdot |I|^{O(1)})$ berechenbar sein
(f und g sind berechenbare Funktionen)

Parametrisierte Reduktionen

Reduktion von Problem \mathcal{L} zu Problem \mathcal{L}'

- bildet jede Instanz (I, k) von \mathcal{L} auf eine Instanz (I', k') von \mathcal{L}' ab
- (I, k) ist ja-Instanz $\Leftrightarrow (I', k')$ ist ja-Instanz; mit $k' \leq g(k)$
- die Abbildung muss in FPT-Zeit $(f(k) \cdot |I|^{O(1)})$ berechenbar sein
(f und g sind berechenbare Funktionen)

Folgerung

- FPT-Algorithmus für $\mathcal{L}' \Rightarrow$ FPT Algorithmus für \mathcal{L}

Parametrisierte Reduktionen

Reduktion von Problem \mathcal{L} zu Problem \mathcal{L}'

- bildet jede Instanz (I, k) von \mathcal{L} auf eine Instanz (I', k') von \mathcal{L}' ab
- (I, k) ist ja-Instanz $\Leftrightarrow (I', k')$ ist ja-Instanz; mit $k' \leq g(k)$
- die Abbildung muss in FPT-Zeit $(f(k) \cdot |I|^{O(1)})$ berechenbar sein
(f und g sind berechenbare Funktionen)

Folgerung

- FPT-Algorithmus für $\mathcal{L}' \Rightarrow$ FPT Algorithmus für \mathcal{L}
- mein Problem \mathcal{L}' effizient lösbar \Rightarrow bekanntes schwieriges Problem \mathcal{L} effizient lösbar
(zumindest, wenn „effizient“ = „FPT“)

Parametrisierte Reduktionen

Reduktion von Problem \mathcal{L} zu Problem \mathcal{L}'

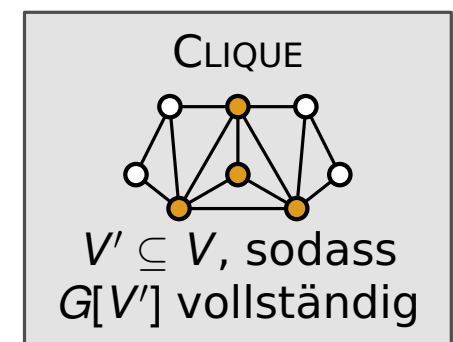
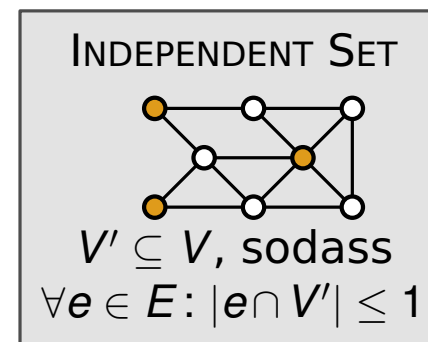
- bildet jede Instanz (I, k) von \mathcal{L} auf eine Instanz (I', k') von \mathcal{L}' ab
- (I, k) ist ja-Instanz $\Leftrightarrow (I', k')$ ist ja-Instanz; mit $k' \leq g(k)$
- die Abbildung muss in FPT-Zeit $(f(k) \cdot |I|^{O(1)})$ berechenbar sein
(f und g sind berechenbare Funktionen)

Folgerung

- FPT-Algorithmus für $\mathcal{L}' \Rightarrow$ FPT Algorithmus für \mathcal{L}
- mein Problem \mathcal{L}' effizient lösbar \Rightarrow bekanntes schwieriges Problem \mathcal{L} effizient lösbar
(zumindest, wenn „effizient“ = „FPT“)

Beispiel

- reduziere INDEPENDENT SET auf CLIQUE



Parametrisierte Reduktionen

Reduktion von Problem \mathcal{L} zu Problem \mathcal{L}'

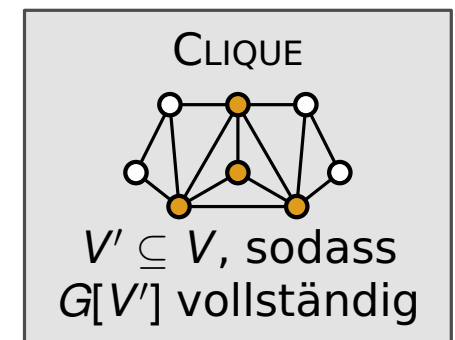
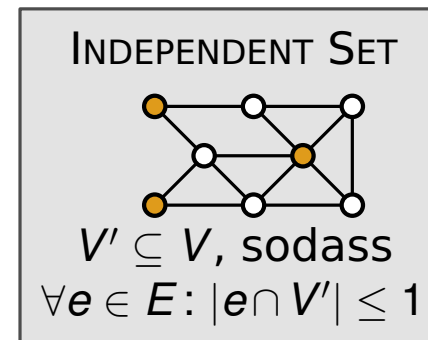
- bildet jede Instanz (I, k) von \mathcal{L} auf eine Instanz (I', k') von \mathcal{L}' ab
- (I, k) ist ja-Instanz $\Leftrightarrow (I', k')$ ist ja-Instanz; mit $k' \leq g(k)$
- die Abbildung muss in FPT-Zeit $(f(k) \cdot |I|^{O(1)})$ berechenbar sein
(f und g sind berechenbare Funktionen)

Folgerung

- FPT-Algorithmus für $\mathcal{L}' \Rightarrow$ FPT Algorithmus für \mathcal{L}
- mein Problem \mathcal{L}' effizient lösbar \Rightarrow bekanntes schwieriges Problem \mathcal{L} effizient lösbar
(zumindest, wenn „effizient“ = „FPT“)

Beispiel

- reduziere INDEPENDENT SET auf CLIQUE
- G hat IS der Größe $k \Leftrightarrow$ Komplement von G hat Clique der Größe k



Parametrisierte Reduktionen

Reduktion von Problem \mathcal{L} zu Problem \mathcal{L}'

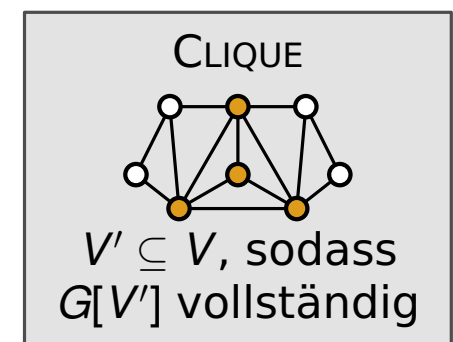
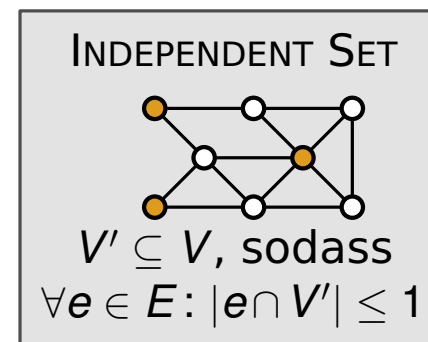
- bildet jede Instanz (I, k) von \mathcal{L} auf eine Instanz (I', k') von \mathcal{L}' ab
- (I, k) ist ja-Instanz $\Leftrightarrow (I', k')$ ist ja-Instanz; mit $k' \leq g(k)$
- die Abbildung muss in FPT-Zeit $(f(k) \cdot |I|^{O(1)})$ berechenbar sein
(f und g sind berechenbare Funktionen)

Folgerung

- FPT-Algorithmus für $\mathcal{L}' \Rightarrow$ FPT Algorithmus für \mathcal{L}
- mein Problem \mathcal{L}' effizient lösbar \Rightarrow bekanntes schwieriges Problem \mathcal{L} effizient lösbar
(zumindest, wenn „effizient“ = „FPT“)

Beispiel

- reduziere INDEPENDENT SET auf CLIQUE
- G hat IS der Größe $k \Leftrightarrow$ Komplement von G hat Clique der Größe k
- also: CLIQUE \in FPT \Rightarrow IS \in FPT



Parametrisierte Reduktionen

Reduktion von Problem \mathcal{L} zu Problem \mathcal{L}'

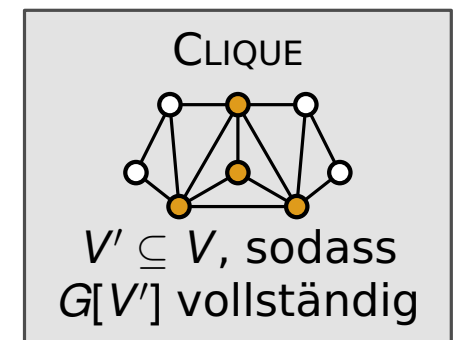
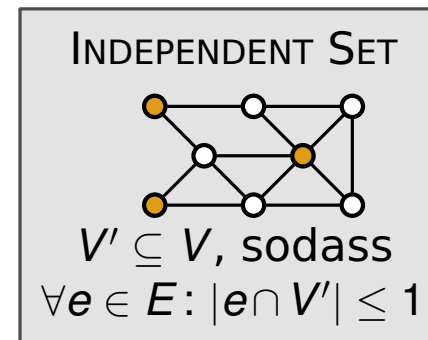
- bildet jede Instanz (I, k) von \mathcal{L} auf eine Instanz (I', k') von \mathcal{L}' ab
- (I, k) ist ja-Instanz $\Leftrightarrow (I', k')$ ist ja-Instanz; mit $k' \leq g(k)$
- die Abbildung muss in FPT-Zeit $(f(k) \cdot |I|^{O(1)})$ berechenbar sein
(f und g sind berechenbare Funktionen)

Folgerung

- FPT-Algorithmus für $\mathcal{L}' \Rightarrow$ FPT Algorithmus für \mathcal{L}
- mein Problem \mathcal{L}' effizient lösbar \Rightarrow bekanntes schwieriges Problem \mathcal{L} effizient lösbar
(zumindest, wenn „effizient“ = „FPT“)

Beispiel

- reduziere INDEPENDENT SET auf CLIQUE
- G hat IS der Größe $k \Leftrightarrow$ Komplement von G hat Clique der Größe k
- also: CLIQUE \in FPT \Rightarrow IS \in FPT
- Umkehrung gilt auch, also:
CLIQUE \in FPT \Leftrightarrow IS \in FPT



Parametrisierte Reduktionen

Reduktion von Problem \mathcal{L} zu Problem \mathcal{L}'

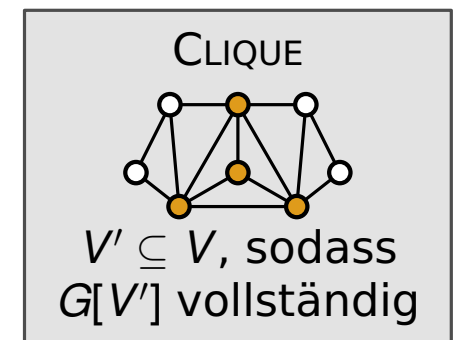
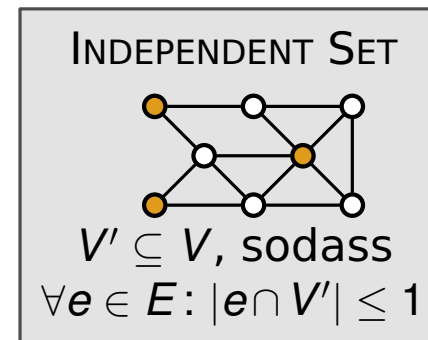
- bildet jede Instanz (I, k) von \mathcal{L} auf eine Instanz (I', k') von \mathcal{L}' ab
- (I, k) ist ja-Instanz $\Leftrightarrow (I', k')$ ist ja-Instanz; mit $k' \leq g(k)$
- die Abbildung muss in FPT-Zeit $(f(k) \cdot |I|^{O(1)})$ berechenbar sein
(f und g sind berechenbare Funktionen)

Folgerung

- FPT-Algorithmus für $\mathcal{L}' \Rightarrow$ FPT Algorithmus für \mathcal{L}
- mein Problem \mathcal{L}' effizient lösbar \Rightarrow bekanntes schwieriges Problem \mathcal{L} effizient lösbar
(zumindest, wenn „effizient“ = „FPT“)

Beispiel

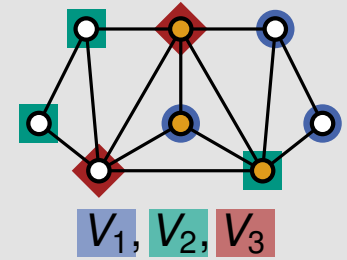
- reduziere INDEPENDENT SET auf CLIQUE
- G hat IS der Größe $k \Leftrightarrow$ Komplement von G hat Clique der Größe k
- also: CLIQUE \in FPT \Rightarrow IS \in FPT
- Umkehrung gilt auch, also:
CLIQUE \in FPT \Leftrightarrow IS \in FPT
- Vermutung: CLIQUE, IS \notin FPT



Bunte Cliques

Problem: MULTICOLORED CLIQUE

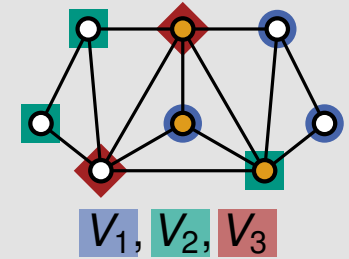
Sei $G = (V, E)$ ein Graph, k ein Parameter und (V_1, \dots, V_k) eine Partitionierung der Knoten. Gibt es eine Clique $V' \subseteq V$ der Größe k , sodass $|V' \cap V_i| = 1$ für alle i ?



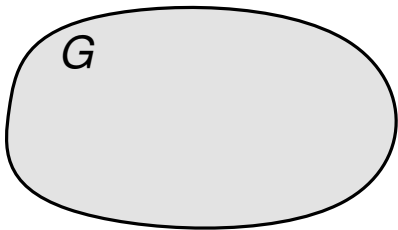
Bunte Cliques

Problem: MULTICOLORED CLIQUE

Sei $G = (V, E)$ ein Graph, k ein Parameter und (V_1, \dots, V_k) eine Partitionierung der Knoten. Gibt es eine Clique $V' \subseteq V$ der Größe k , sodass $|V' \cap V_i| = 1$ für alle i ?



Reduktion: von CLIQUE auf MULTICOLORED CLIQUE

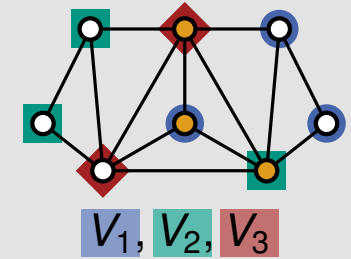


Instanz $(G, 3)$ von
 CLIQUE

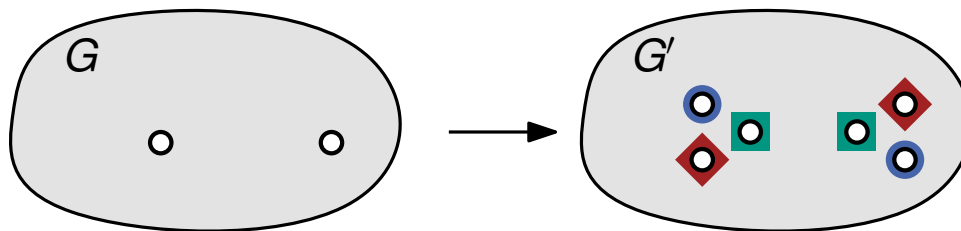
Bunte Cliques

Problem: MULTICOLORED CLIQUE

Sei $G = (V, E)$ ein Graph, k ein Parameter und (V_1, \dots, V_k) ein Partitionierung der Knoten. Gibt es eine Clique $V' \subseteq V$ der Größe k , sodass $|V' \cap V_i| = 1$ für alle i ?



Reduktion: von CLIQUE auf MULTICOLORED CLIQUE



Instanz $(G, 3)$ von
CLIQUE

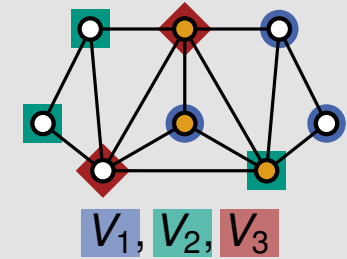
Instanz von MC CLIQUE:
 $(G', 3, (V_1, V_2, V_3))$

- ersetze $v \in V$ durch v^1, \dots, v^k mit $v^i \in V_i$

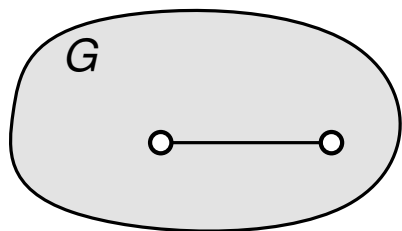
Bunte Cliques

Problem: MULTICOLORED CLIQUE

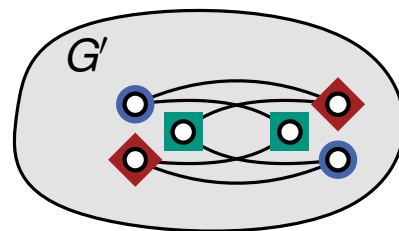
Sei $G = (V, E)$ ein Graph, k ein Parameter und (V_1, \dots, V_k) eine Partitionierung der Knoten. Gibt es eine Clique $V' \subseteq V$ der Größe k , sodass $|V' \cap V_i| = 1$ für alle i ?



Reduktion: von CLIQUE auf MULTICOLORED CLIQUE



Instanz $(G, 3)$ von
CLIQUE



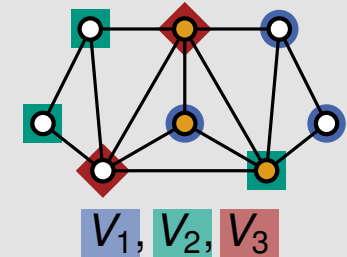
Instanz von MC CLIQUE:
 $(G', 3, (V_1, V_2, V_3))$

- ersetze $v \in V$ durch v^1, \dots, v^k mit $v^i \in V_i$
- für $uv \in E$ verbinde u^i mit v^j für alle $i \neq j$
- $k = k'$

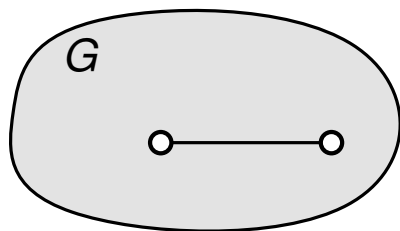
Bunte Cliques

Problem: MULTICOLORED CLIQUE

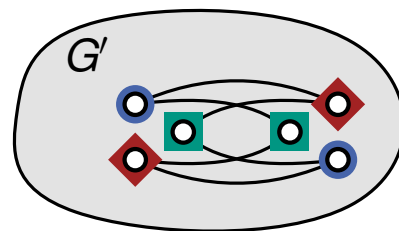
Sei $G = (V, E)$ ein Graph, k ein Parameter und (V_1, \dots, V_k) eine Partitionierung der Knoten. Gibt es eine Clique $V' \subseteq V$ der Größe k , sodass $|V' \cap V_i| = 1$ für alle i ?



Reduktion: von CLIQUE auf MULTICOLORED CLIQUE



Instanz $(G, 3)$ von
CLIQUE



Instanz von MC CLIQUE:
 $(G', 3, (V_1, V_2, V_3))$

- ersetze $v \in V$ durch v^1, \dots, v^k mit $v^i \in V_i$
- für $uv \in E$ verbinde u^i mit v^j für alle $i \neq j$
- $k = k'$

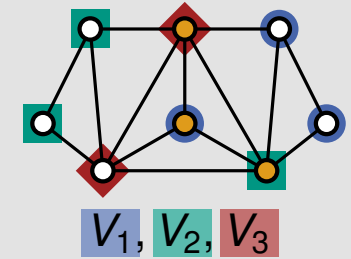
G hat Clique der Größe $k \Rightarrow G'$ hat bunte Clique der Größe k

- sei v_1, \dots, v_k eine Clique in G
- dann ist v_1^1, \dots, v_k^k eine Clique in G'

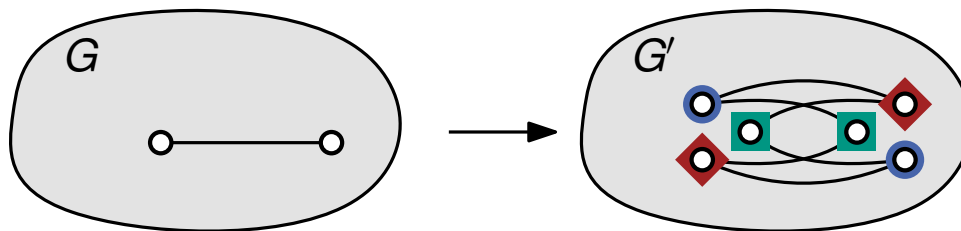
Bunte Cliques

Problem: MULTICOLORED CLIQUE

Sei $G = (V, E)$ ein Graph, k ein Parameter und (V_1, \dots, V_k) eine Partitionierung der Knoten. Gibt es eine Clique $V' \subseteq V$ der Größe k , sodass $|V' \cap V_i| = 1$ für alle i ?



Reduktion: von CLIQUE auf MULTICOLORED CLIQUE



Instanz $(G, 3)$ von
CLIQUE

Instanz von MC CLIQUE:
 $(G', 3, (V_1, V_2, V_3))$

- ersetze $v \in V$ durch v^1, \dots, v^k mit $v^i \in V_i$
- für $uv \in E$ verbinde u^i mit v^j für alle $i \neq j$
- $k = k'$

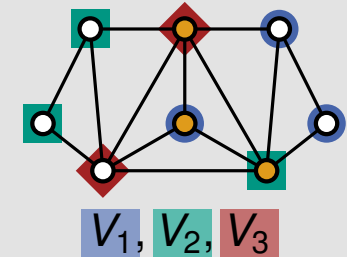
G hat Clique der Größe $k \Rightarrow G'$ hat bunte Clique der Größe k

- sei v_1, \dots, v_k eine Clique in G
- dann ist v_1^1, \dots, v_k^k eine Clique in G'
- alle Knoten haben unterschiedliche Farben

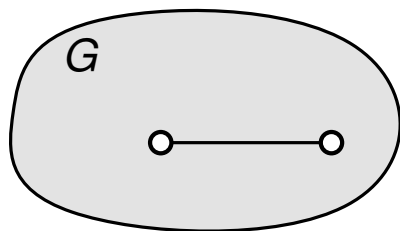
Bunte Cliques

Problem: MULTICOLORED CLIQUE

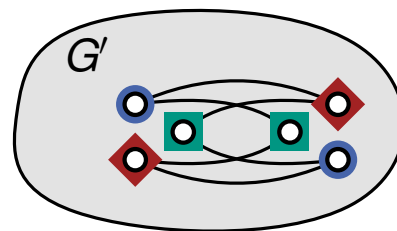
Sei $G = (V, E)$ ein Graph, k ein Parameter und (V_1, \dots, V_k) eine Partitionierung der Knoten. Gibt es eine Clique $V' \subseteq V$ der Größe k , sodass $|V' \cap V_i| = 1$ für alle i ?



Reduktion: von CLIQUE auf MULTICOLORED CLIQUE



Instanz $(G, 3)$ von
CLIQUE



Instanz von MC CLIQUE:
 $(G', 3, (V_1, V_2, V_3))$

- ersetze $v \in V$ durch v^1, \dots, v^k mit $v^i \in V_i$
- für $uv \in E$ verbinde u^i mit v^j für alle $i \neq j$
- $k = k'$

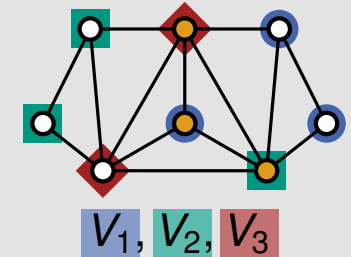
G' hat bunte Clique der Größe $k \Rightarrow G$ hat Clique der Größe k

- sei $v_{\pi(1)}^1, \dots, v_{\pi(k)}^k$ bunte Clique in G mit $\pi: \{1, \dots, k\} \rightarrow \{1, \dots, n\}$

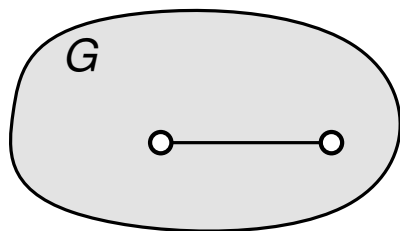
Bunte Cliques

Problem: MULTICOLORED CLIQUE

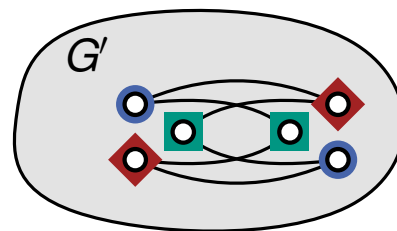
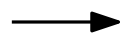
Sei $G = (V, E)$ ein Graph, k ein Parameter und (V_1, \dots, V_k) eine Partitionierung der Knoten. Gibt es eine Clique $V' \subseteq V$ der Größe k , sodass $|V' \cap V_i| = 1$ für alle i ?



Reduktion: von CLIQUE auf MULTICOLORED CLIQUE



Instanz $(G, 3)$ von CLIQUE



Instanz von MC CLIQUE:
 $(G', 3, (V_1, V_2, V_3))$

- ersetze $v \in V$ durch v^1, \dots, v^k mit $v^i \in V_i$
- für $uv \in E$ verbinde u^i mit v^j für alle $i \neq j$
- $k = k'$

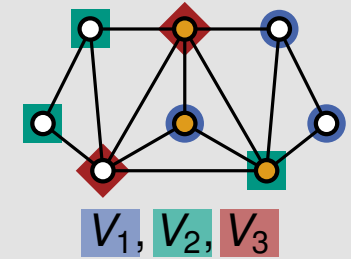
G' hat bunte Clique der Größe $k \Rightarrow G$ hat Clique der Größe k

- sei $v_{\pi(1)}^1, \dots, v_{\pi(k)}^k$ bunte Clique in G mit $\pi: \{1, \dots, k\} \rightarrow \{1, \dots, n\}$
- π ist injektiv (die bunte Clique enthält keine zwei Kopien desselben Knotens)

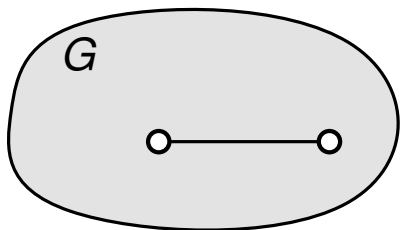
Bunte Cliques

Problem: MULTICOLORED CLIQUE

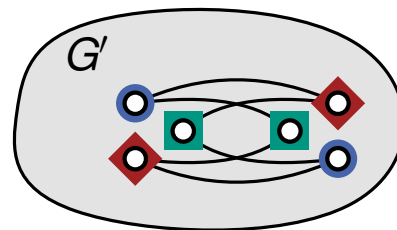
Sei $G = (V, E)$ ein Graph, k ein Parameter und (V_1, \dots, V_k) ein Partitionierung der Knoten. Gibt es eine Clique $V' \subseteq V$ der Größe k , sodass $|V' \cap V_i| = 1$ für alle i ?



Reduktion: von CLIQUE auf MULTICOLORED CLIQUE



Instanz $(G, 3)$ von CLIQUE



Instanz von MC CLIQUE:
 $(G', 3, (V_1, V_2, V_3))$

- ersetze $v \in V$ durch v^1, \dots, v^k mit $v^i \in V_i$
- für $uv \in E$ verbinde u^i mit v^j für alle $i \neq j$
- $k = k'$

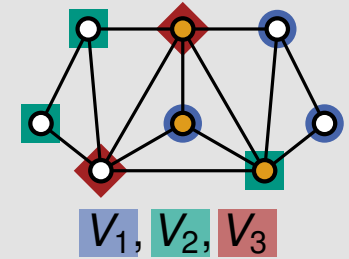
G' hat bunte Clique der Größe $k \Rightarrow G$ hat Clique der Größe k

- sei $v_{\pi(1)}^1, \dots, v_{\pi(k)}^k$ bunte Clique in G mit $\pi: \{1, \dots, k\} \rightarrow \{1, \dots, n\}$
- π ist injektiv (die bunte Clique enthält keine zwei Kopien desselben Knotens)
- damit sind $v_{\pi(1)}, \dots, v_{\pi(k)}$ k unterschiedliche Knoten, die in G eine Clique bilden

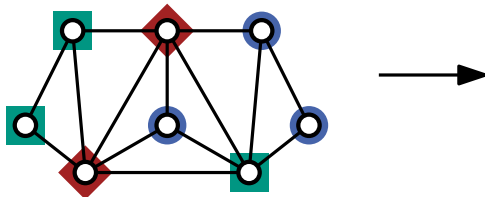
Bunte Cliques

Problem: MULTICOLORED CLIQUE

Sei $G = (V, E)$ ein Graph, k ein Parameter und (V_1, \dots, V_k) ein Partitionierung der Knoten. Gibt es eine Clique $V' \subseteq V$ der Größe k , sodass $|V' \cap V_i| = 1$ für alle i ?



Reduktion: von MULTICOLORED CLIQUE auf CLIQUE



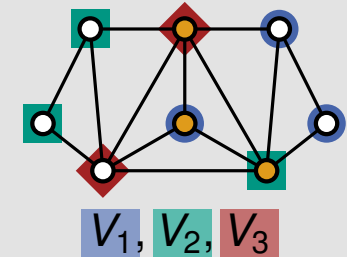
Instanz von MC CLIQUE:

$(G, 3, (V_1, V_2, V_3))$

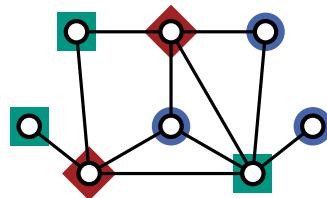
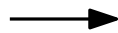
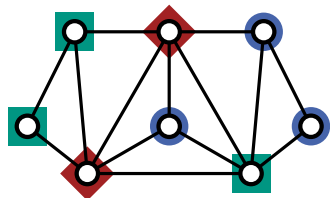
Bunte Cliques

Problem: MULTICOLORED CLIQUE

Sei $G = (V, E)$ ein Graph, k ein Parameter und (V_1, \dots, V_k) ein Partitionierung der Knoten. Gibt es eine Clique $V' \subseteq V$ der Größe k , sodass $|V' \cap V_i| = 1$ für alle i ?



Reduktion: von MULTICOLORED CLIQUE auf CLIQUE



- lösche Kanten zwischen gleichfarbigen Knoten
- $k' = k$

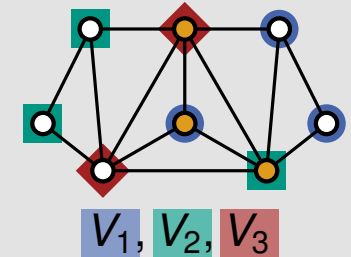
Instanz von MC CLIQUE:
 $(G, 3, (V_1, V_2, V_3))$

Instanz $(G', 3)$
 von CLIQUE

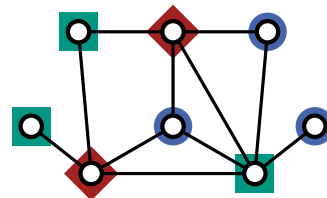
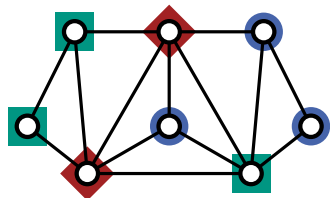
Bunte Cliques

Problem: MULTICOLORED CLIQUE

Sei $G = (V, E)$ ein Graph, k ein Parameter und (V_1, \dots, V_k) ein Partitionierung der Knoten. Gibt es eine Clique $V' \subseteq V$ der Größe k , sodass $|V' \cap V_i| = 1$ für alle i ?



Reduktion: von MULTICOLORED CLIQUE auf CLIQUE



- lösche Kanten zwischen gleichfarbigen Knoten
- $k' = k$

Instanz von MC CLIQUE:
 $(G, 3, (V_1, V_2, V_3))$

Instanz $(G', 3)$
 von CLIQUE

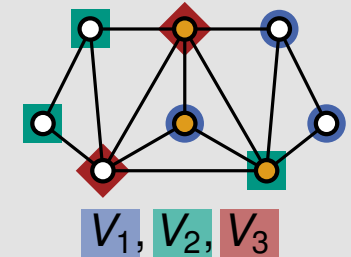
G hat bunte Clique der Größe $k \Rightarrow G'$ hat Clique der Größe k

- aus der bunten Clique wurden keine Kanten gelöscht
- damit bleibt sie eine Clique der Größe k in G'

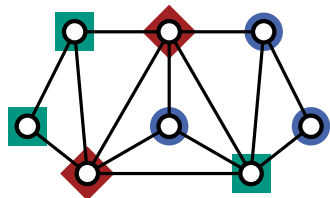
Bunte Cliques

Problem: MULTICOLORED CLIQUE

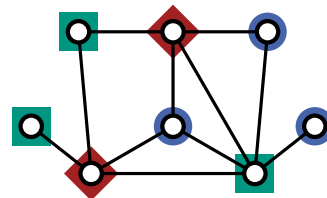
Sei $G = (V, E)$ ein Graph, k ein Parameter und (V_1, \dots, V_k) ein Partitionierung der Knoten. Gibt es eine Clique $V' \subseteq V$ der Größe k , sodass $|V' \cap V_i| = 1$ für alle i ?



Reduktion: von MULTICOLORED CLIQUE auf CLIQUE



Instanz von MC CLIQUE:
 $(G, 3, (V_1, V_2, V_3))$



Instanz $(G', 3)$
 von CLIQUE

- lösche Kanten zwischen gleichfarbigen Knoten
- $k' = k$

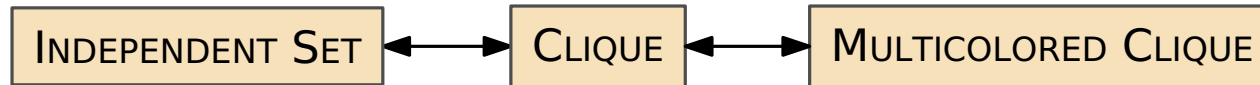
G hat bunte Clique der Größe $k \Rightarrow G'$ hat Clique der Größe k

- aus der bunten Clique wurden keine Kanten gelöscht
- damit bleibt sie eine Clique der Größe k in G'

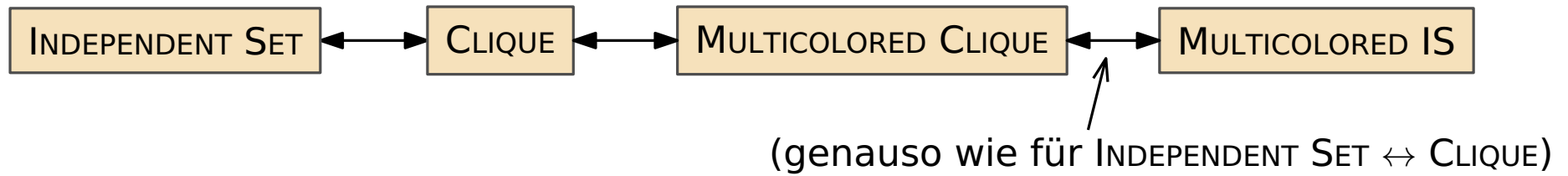
G' hat Clique der Größe $k \Rightarrow G$ hat bunte Clique der Größe k

- in G' gibt es keine adjazente Knoten der gleichen Farbe
- jede Clique muss bunt sein

Bisherige FPT-Reduktionen



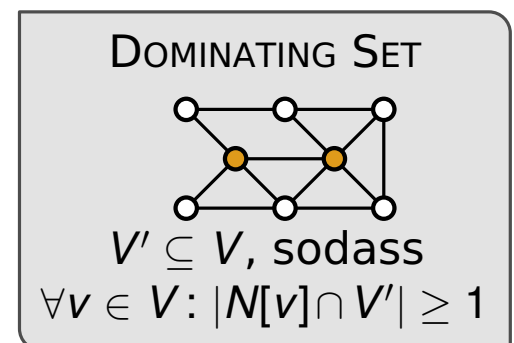
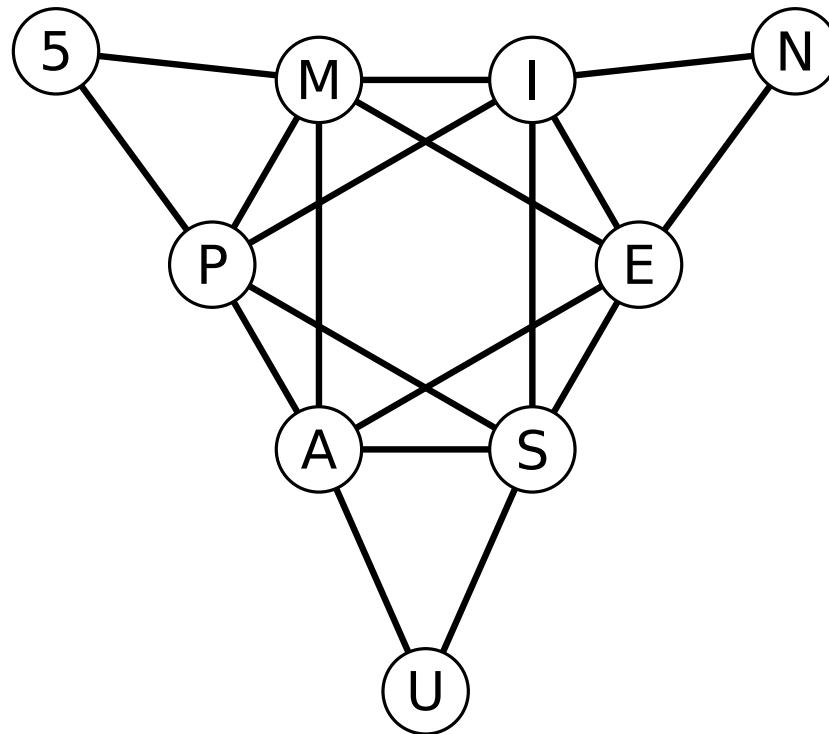
Bisherige FPT-Reduktionen



Dominating Set

Ziel

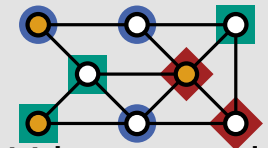
- wähle möglichst wenige Knoten aus
- für jeden Knoten gilt: er selbst oder ein Nachbar ist ausgewählt



DOMINATING SET

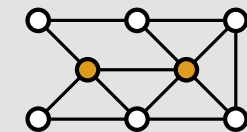
Reduziere MULTICOLORED INDEPENDENT SET auf DOMINATING SET

MC INDEPENDENT SET



$V' \subseteq V$ bunt, sodass
 $\forall e \in E: |e \cap V'| \leq 1$

DOMINATING SET



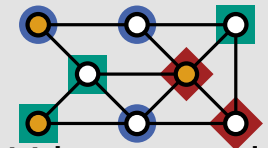
$V' \subseteq V$, sodass
 $\forall v \in V: |N[v] \cap V'| \geq 1$

DOMINATING SET

Reduziere MULTICOLORED INDEPENDENT SET auf DOMINATING SET

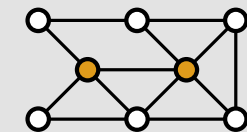
- erzwingen folgende Eigenschaften mittels DOMINATING SET Instanzen
 - erzwinge die Wahl genau eines Elements für jede Farbklasse
 - verbiete die gleichzeitige Wahl von gewissen Knotenpaaren

MC INDEPENDENT SET



$V' \subseteq V$ bunt, sodass
 $\forall e \in E: |e \cap V'| \leq 1$

DOMINATING SET



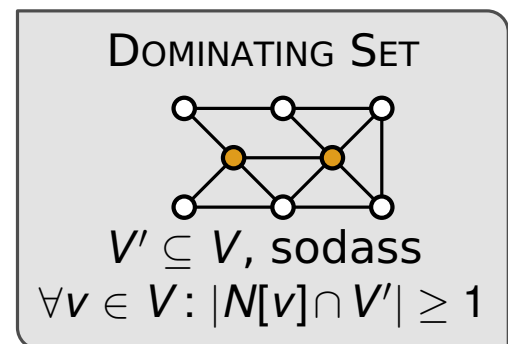
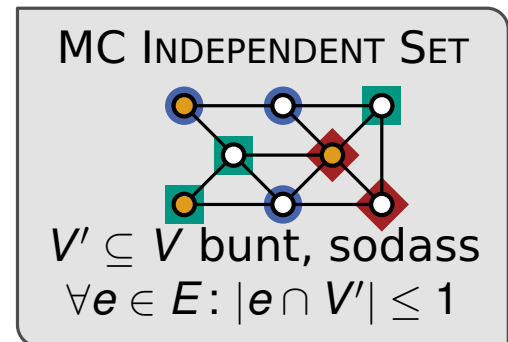
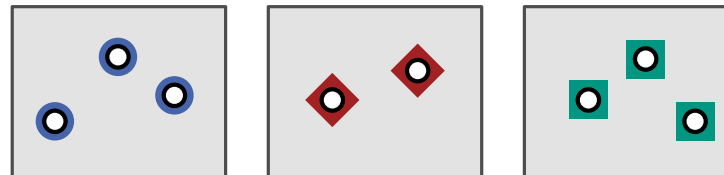
$V' \subseteq V$, sodass
 $\forall v \in V: |N[v] \cap V'| \geq 1$

DOMINATING SET

Reduziere MULTICOLORED INDEPENDENT SET auf DOMINATING SET

- erzwingen folgende Eigenschaften mittels DOMINATING SET Instanzen
 - erzwinge die Wahl genau eines Elements für jede Farbklasse
 - verbiete die gleichzeitige Wahl von gewissen Knotenpaaren

Ein Element aus jeder Farbklasse



DOMINATING SET

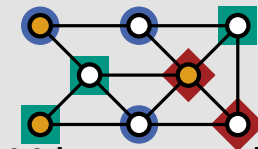
Reduziere MULTICOLORED INDEPENDENT SET auf DOMINATING SET

- erzwingen folgende Eigenschaften mittels DOMINATING SET Instanzen
 - erzwinge die Wahl genau eines Elements für jede Farbklasse
 - verbiete die gleichzeitige Wahl von gewissen Knotenpaaren

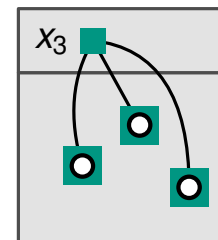
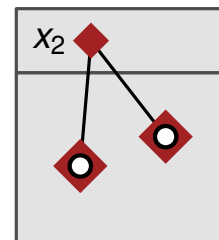
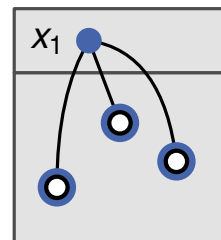
Ein Element aus jeder Farbklasse

- Knoten x_i mit Kanten zu allen Knoten in V_i und ohne weitere Kanten
 (erzwingt die Wahl mindestens eines Knotens aus $V_i \cup \{x_i\}$)

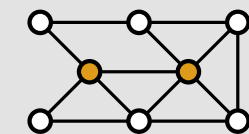
MC INDEPENDENT SET



$V' \subseteq V$ bunt, sodass
 $\forall e \in E: |e \cap V'| \leq 1$



DOMINATING SET



$V' \subseteq V$, sodass
 $\forall v \in V: |N[v] \cap V'| \geq 1$

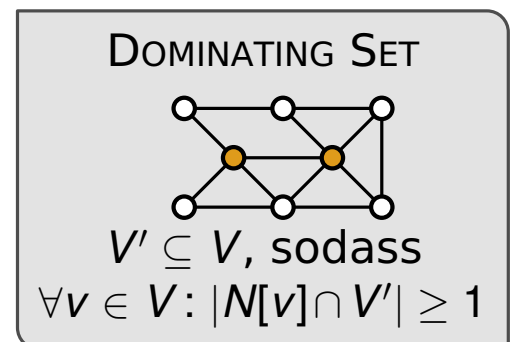
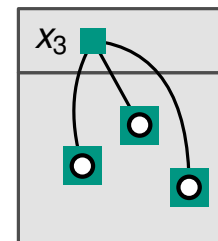
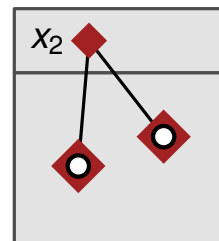
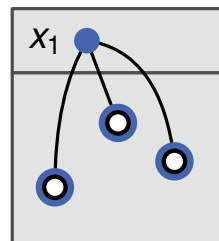
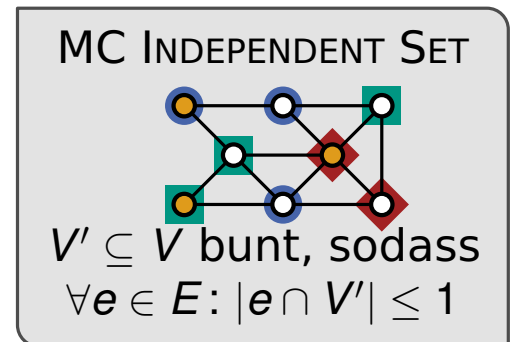
DOMINATING SET

Reduziere MULTICOLORED INDEPENDENT SET auf DOMINATING SET

- erzwingen folgende Eigenschaften mittels DOMINATING SET Instanzen
 - erzwinge die Wahl genau eines Elements für jede Farbklasse
 - verbiete die gleichzeitige Wahl von gewissen Knotenpaaren

Ein Element aus jeder Farbklasse

- Knoten x_i mit Kanten zu allen Knoten in V_i und ohne weitere Kanten
(erzwingt die Wahl mindestens eines Knotens aus $V_i \cup \{x_i\}$)
- Problem: man könnte x_i wählen



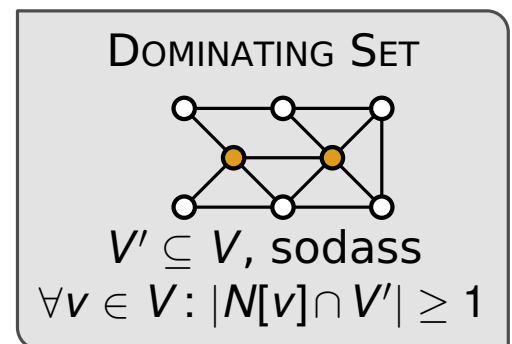
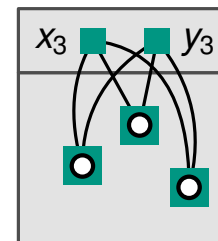
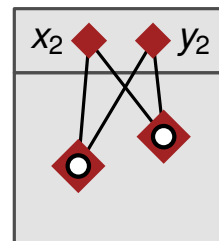
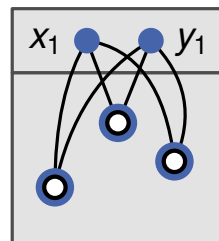
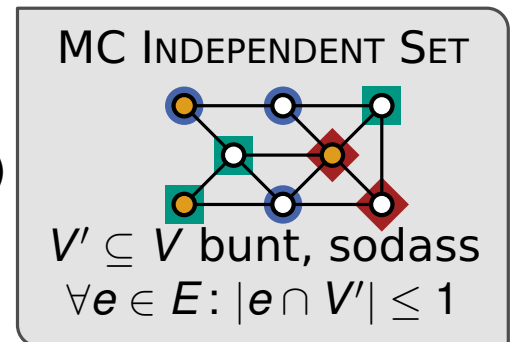
DOMINATING SET

Reduziere MULTICOLORED INDEPENDENT SET auf DOMINATING SET

- erzwingen folgende Eigenschaften mittels DOMINATING SET Instanzen
 - erzwinge die Wahl genau eines Elements für jede Farbkategorie
 - verbiete die gleichzeitige Wahl von gewissen Knotenpaaren

Ein Element aus jeder Farbkategorie

- Knoten x_i mit Kanten zu allen Knoten in V_i und ohne weitere Kanten
(erzwingt die Wahl mindestens eines Knotens aus $V_i \cup \{x_i\}$)
- Problem: man könnte x_i wählen
- Lösung: erstelle weiteren Knoten y_i genauso
(x_i und y_i zu wählen ist zu teuer für ein DS der Größe k)



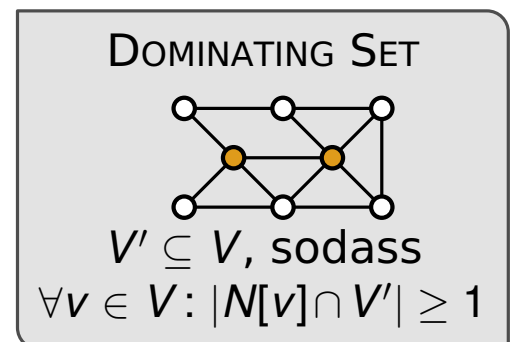
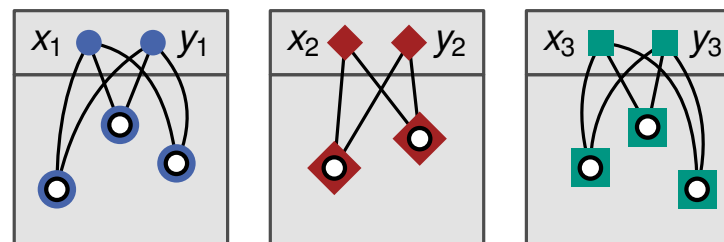
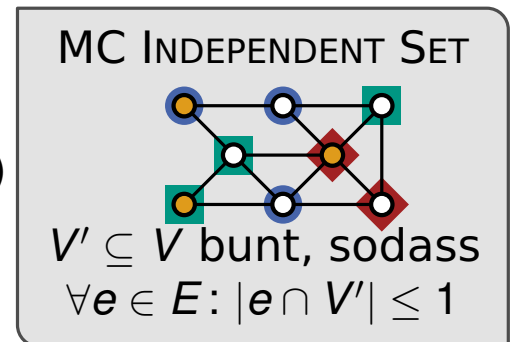
DOMINATING SET

Reduziere MULTICOLORED INDEPENDENT SET auf DOMINATING SET

- erzwingen folgende Eigenschaften mittels DOMINATING SET Instanzen
 - erzwinge die Wahl genau eines Elements für jede Farbkategorie
 - verbiete die gleichzeitige Wahl von gewissen Knotenpaaren

Ein Element aus jeder Farbkategorie

- Knoten x_i mit Kanten zu allen Knoten in V_i und ohne weitere Kanten
(erzwingt die Wahl mindestens eines Knotens aus $V_i \cup \{x_i\}$)
- Problem: man könnte x_i wählen
- Lösung: erstelle weiteren Knoten y_i genauso
(x_i und y_i zu wählen ist zu teuer für ein DS der Größe k)
- damit es genügt einen Knoten aus V_i zu wählen:



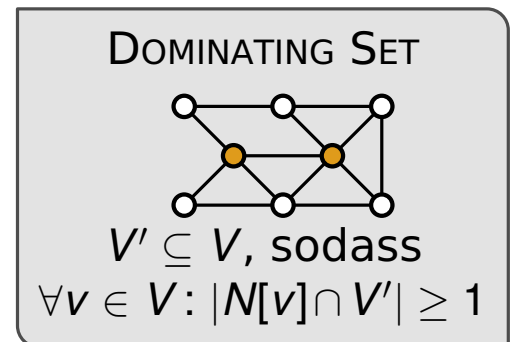
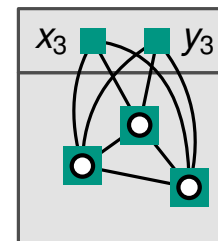
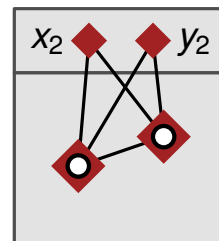
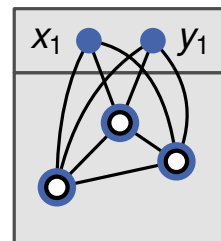
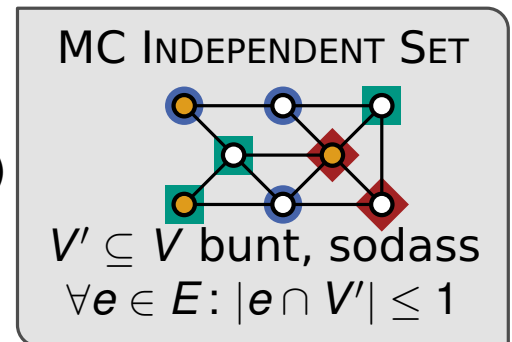
DOMINATING SET

Reduziere MULTICOLORED INDEPENDENT SET auf DOMINATING SET

- erzwingen folgende Eigenschaften mittels DOMINATING SET Instanzen
 - erzwinge die Wahl genau eines Elements für jede Farbkategorie
 - verbiете die gleichzeitige Wahl von gewissen Knotenpaaren

Ein Element aus jeder Farbkategorie

- Knoten x_i mit Kanten zu allen Knoten in V_i und ohne weitere Kanten
(erzwingt die Wahl mindestens eines Knotens aus $V_i \cup \{x_i\}$)
- Problem: man könnte x_i wählen
- Lösung: erstelle weiteren Knoten y_i genauso
(x_i und y_i zu wählen ist zu teuer für ein DS der Größe k)
- damit es genügt einen Knoten aus V_i zu wählen:
erstelle Clique auf diesen Knoten



DOMINATING SET

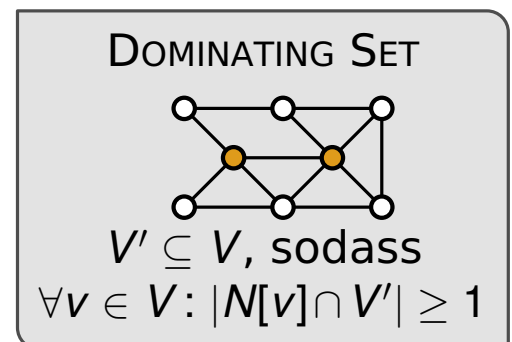
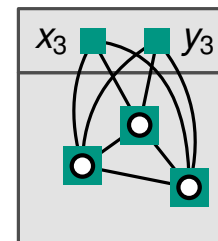
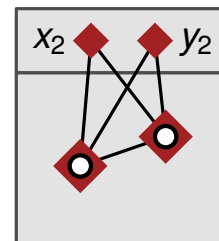
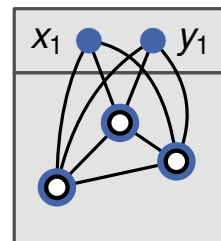
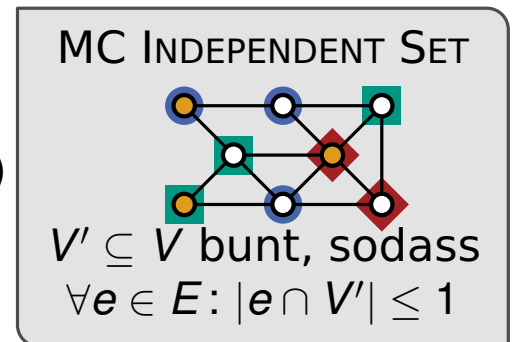
Reduziere MULTICOLORED INDEPENDENT SET auf DOMINATING SET

- erzwingen folgende Eigenschaften mittels DOMINATING SET Instanzen
 - erzwinge die Wahl genau eines Elements für jede Farbklasse
 - verbiете die gleichzeitige Wahl von gewissen Knotenpaaren



Ein Element aus jeder Farbklasse

- Knoten x_i mit Kanten zu allen Knoten in V_i und ohne weitere Kanten
(erzwingt die Wahl mindestens eines Knotens aus $V_i \cup \{x_i\}$)
- Problem: man könnte x_i wählen
- Lösung: erstelle weiteren Knoten y_i genauso
(x_i und y_i zu wählen ist zu teuer für ein DS der Größe k)
- damit es genügt einen Knoten aus V_i zu wählen:
erstelle Clique auf diesen Knoten



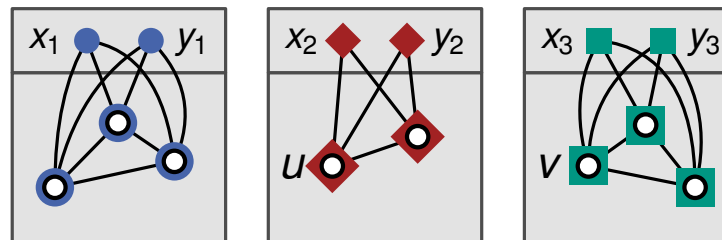
DOMINATING SET

Reduziere MULTICOLORED INDEPENDENT SET auf DOMINATING SET

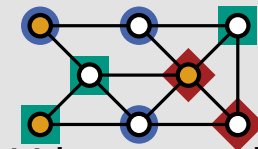
- erzwingen folgende Eigenschaften mittels DOMINATING SET Instanzen
 - erzwinge die Wahl genau eines Elements für jede Farbklasse
 - verbiete die gleichzeitige Wahl von gewissen Knotenpaaren



Verbiete gleichzeitige Wahl von u und v

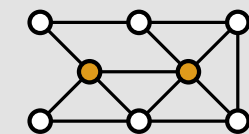


MC INDEPENDENT SET



$V' \subseteq V$ bunt, sodass
 $\forall e \in E: |e \cap V'| \leq 1$

DOMINATING SET



$V' \subseteq V$, sodass
 $\forall v \in V: |N[v] \cap V'| \geq 1$

DOMINATING SET

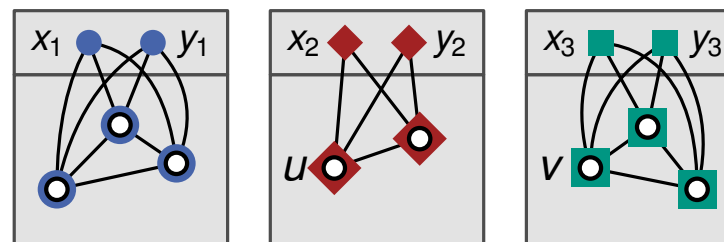
Reduziere MULTICOLORED INDEPENDENT SET auf DOMINATING SET

- erzwingen folgende Eigenschaften mittels DOMINATING SET Instanzen
 - erzwinge die Wahl genau eines Elements für jede Farbklasse
 - verbiete die gleichzeitige Wahl von gewissen Knotenpaaren



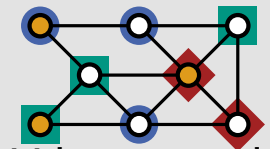
Verbiete gleichzeitige Wahl von u und v

- Idee: füge Knoten z ein, der immer abgedeckt ist, außer wenn u und v ausgewählt wurden



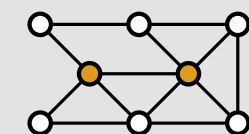
z ●

MC INDEPENDENT SET



$V' \subseteq V$ bunt, sodass
 $\forall e \in E: |e \cap V'| \leq 1$

DOMINATING SET



$V' \subseteq V$, sodass
 $\forall v \in V: |N[v] \cap V'| \geq 1$

DOMINATING SET

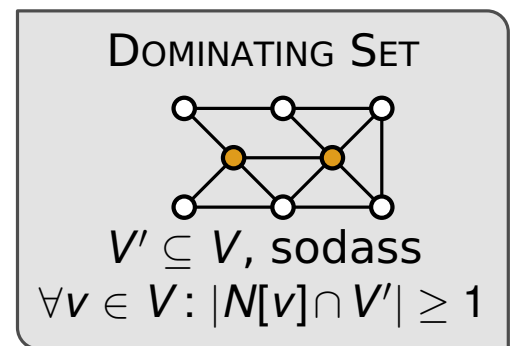
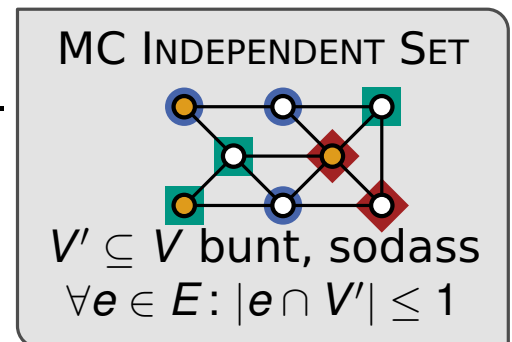
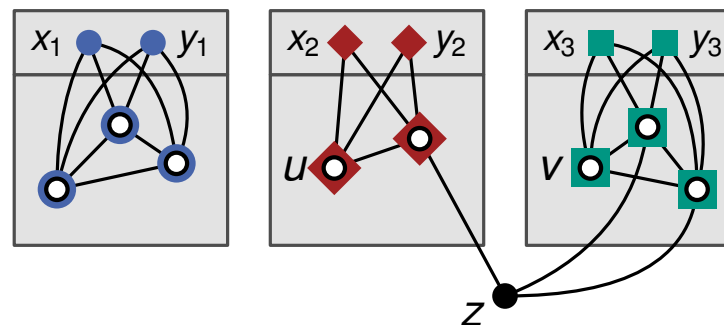
Reduziere MULTICOLORED INDEPENDENT SET auf DOMINATING SET

- erzwingen folgende Eigenschaften mittels DOMINATING SET Instanzen
 - erzwinge die Wahl genau eines Elements für jede Farbklasse
 - verbiete die gleichzeitige Wahl von gewissen Knotenpaaren



Verbiete gleichzeitige Wahl von u und v

- Idee: füge Knoten z ein, der immer abgedeckt ist, außer wenn u und v ausgewählt wurden
 - verbinde z weder mit u noch mit v
 - aber mit allen anderen Knoten aus deren Farb-
klassen



DOMINATING SET

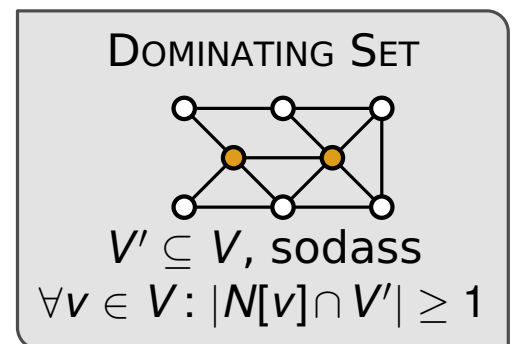
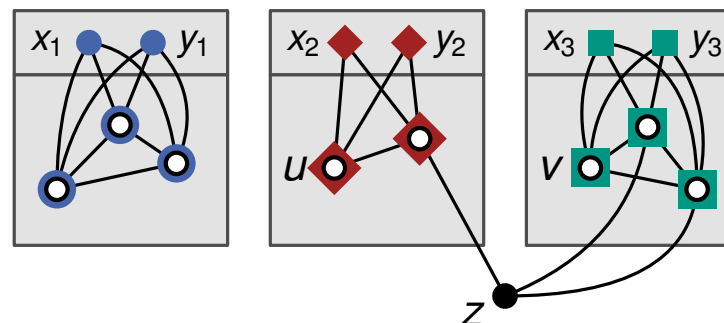
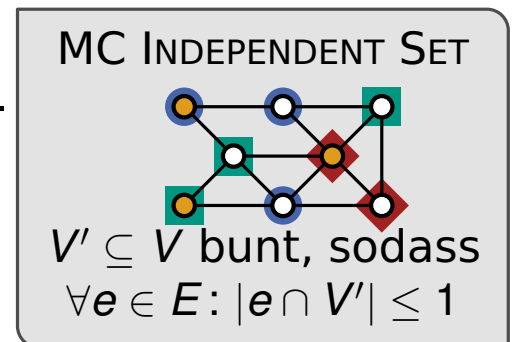
Reduziere MULTICOLORED INDEPENDENT SET auf DOMINATING SET

- erzwingen folgende Eigenschaften mittels DOMINATING SET Instanzen
 - erzwinge die Wahl genau eines Elements für jede Farbklasse
 - verbiete die gleichzeitige Wahl von gewissen Knotenpaaren

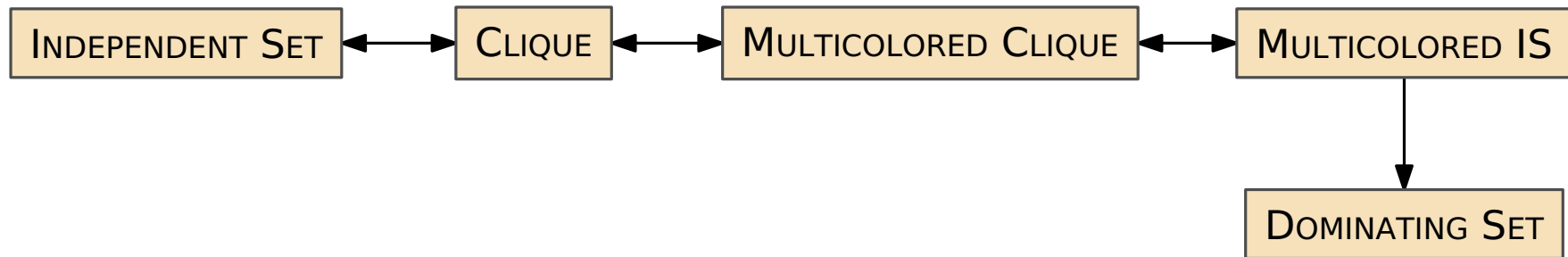


Verbiete gleichzeitige Wahl von u und v

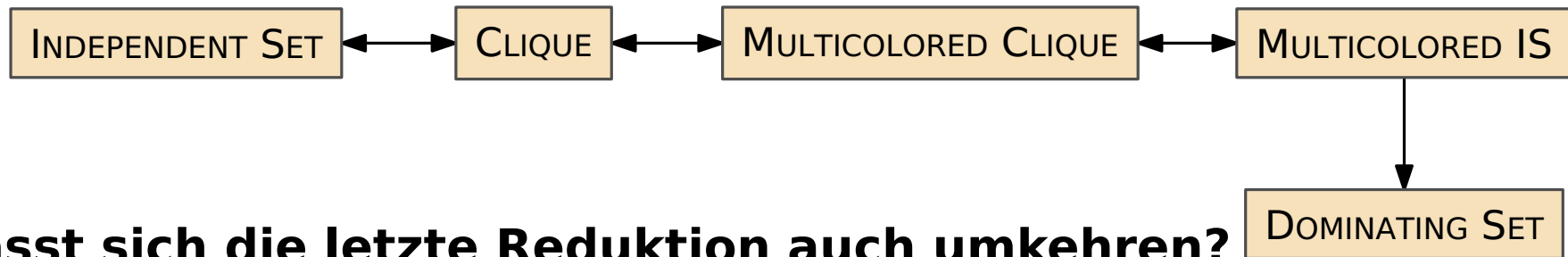
- Idee: füge Knoten z ein, der immer abgedeckt ist, außer wenn u und v ausgewählt wurden
 - verbinde z weder mit u noch mit v
 - aber mit allen anderen Knoten aus deren Farb-
klassen



Bisherige FPT-Reduktionen



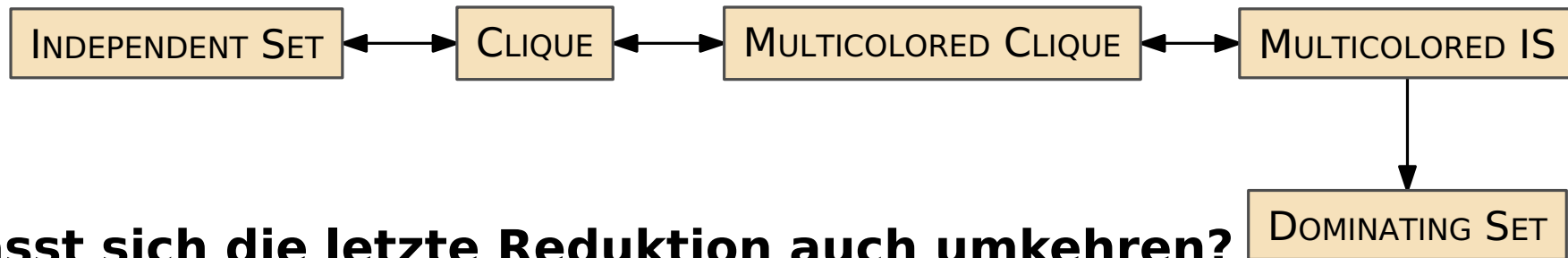
Bisherige FPT-Reduktionen



Lässt sich die letzte Reduktion auch umkehren?

- wissen wir nicht; Vermutung: nicht umkehrbar

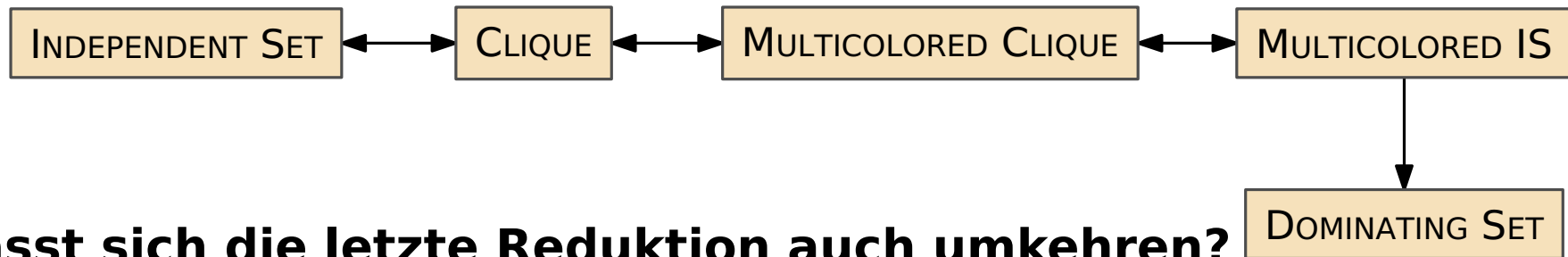
Bisherige FPT-Reduktionen



Lässt sich die letzte Reduktion auch umkehren?

- wissen wir nicht; Vermutung: nicht umkehrbar
- DOMINATING SET ist also (vermutlich) echt schwerer als CLIQUE oder INDEPENDENT SET
 (DS \in FPT \Rightarrow IS \in FPT, aber nicht umgekehrt)

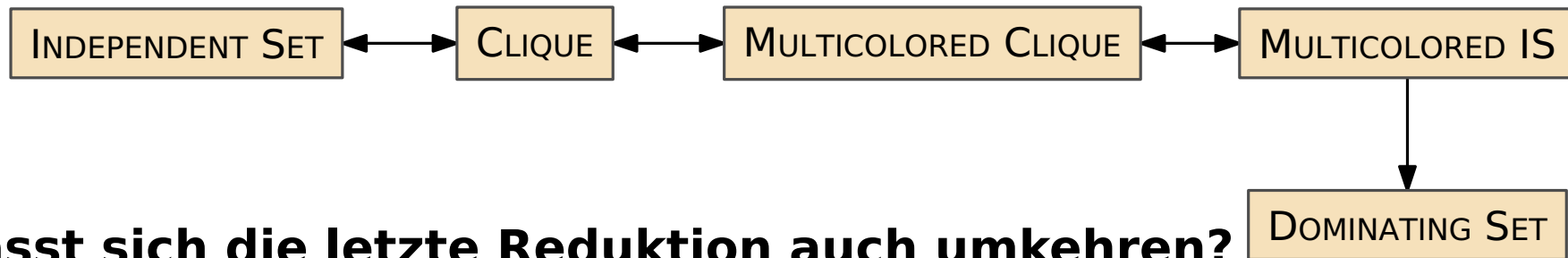
Bisherige FPT-Reduktionen



Lässt sich die letzte Reduktion auch umkehren?

- wissen wir nicht; Vermutung: nicht umkehrbar
- DOMINATING SET ist also (vermutlich) echt schwerer als CLIQUE oder INDEPENDENT SET
($DS \in \text{FPT} \Rightarrow IS \in \text{FPT}$, aber nicht umgekehrt)
- wir brauchen also einen abgestuften Begriff der Schwere

Bisherige FPT-Reduktionen

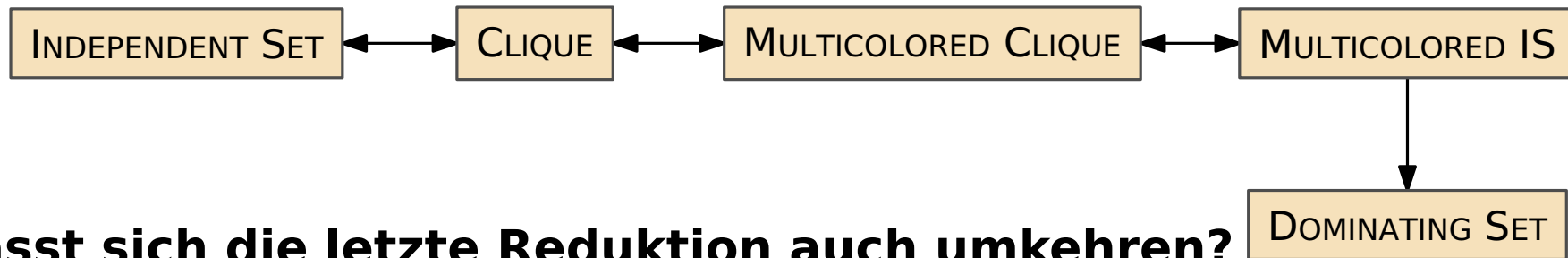


Lässt sich die letzte Reduktion auch umkehren?

- wissen wir nicht; Vermutung: nicht umkehrbar
- DOMINATING SET ist also (vermutlich) echt schwerer als CLIQUE oder INDEPENDENT SET
($DS \in \text{FPT} \Rightarrow IS \in \text{FPT}$, aber nicht umgekehrt)
- wir brauchen also einen abgestuften Begriff der Schwere

Vergleich mit der Komplexitätsklasse P

Bisherige FPT-Reduktionen



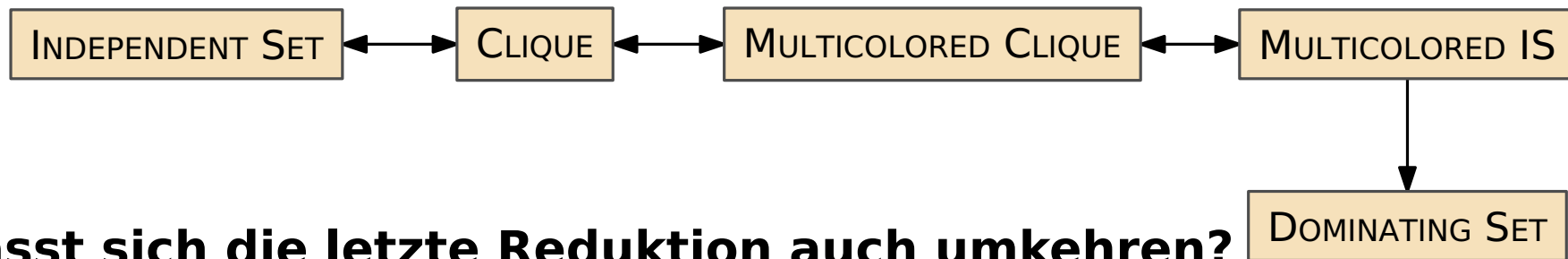
Lässt sich die letzte Reduktion auch umkehren?

- wissen wir nicht; Vermutung: nicht umkehrbar
- DOMINATING SET ist also (vermutlich) echt schwerer als CLIQUE oder INDEPENDENT SET
($DS \in \text{FPT} \Rightarrow IS \in \text{FPT}$, aber nicht umgekehrt)
- wir brauchen also einen abgestuften Begriff der Schwere

Vergleich mit der Komplexitätsklasse P

- hier reicht üblicherweise ein Begriff: NP-Schwere
- es gibt aber auch intermediate-Probleme (angenommen $P \neq \text{NP}$)

Bisherige FPT-Reduktionen



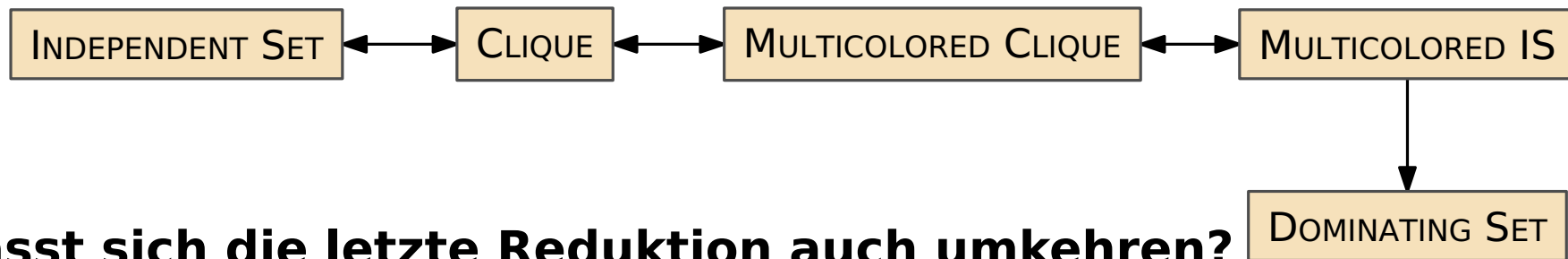
Lässt sich die letzte Reduktion auch umkehren?

- wissen wir nicht; Vermutung: nicht umkehrbar
- DOMINATING SET ist also (vermutlich) echt schwerer als CLIQUE oder INDEPENDENT SET
($DS \in \text{FPT} \Rightarrow IS \in \text{FPT}$, aber nicht umgekehrt)
- wir brauchen also einen abgestuften Begriff der Schwere

Vergleich mit der Komplexitätsklasse P

- hier reicht üblicherweise ein Begriff: NP-Schwere
- es gibt aber auch intermediate-Probleme (angenommen $P \neq \text{NP}$)
- es ist allerdings kein natürliches Problem aus NPI bekannt
(Kandidaten: Primfaktorzerlegung, Graphisomorphie)

Bisherige FPT-Reduktionen



Lässt sich die letzte Reduktion auch umkehren?

- wissen wir nicht; Vermutung: nicht umkehrbar
- DOMINATING SET ist also (vermutlich) echt schwerer als CLIQUE oder INDEPENDENT SET
($DS \in \text{FPT} \Rightarrow IS \in \text{FPT}$, aber nicht umgekehrt)
- wir brauchen also einen abgestuften Begriff der Schwere

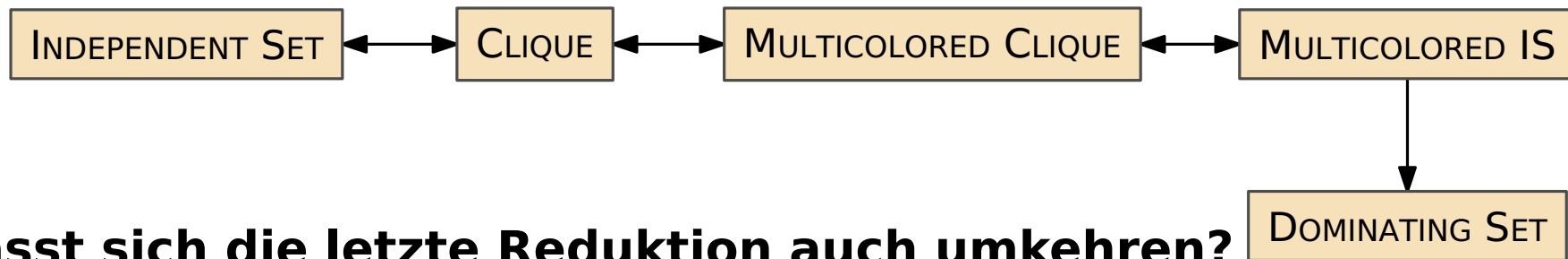
Vergleich mit der Komplexitätsklasse P

- hier reicht üblicherweise ein Begriff: NP-Schwere
- es gibt aber auch intermediate-Probleme (angenommen $P \neq NP$)
- es ist allerdings kein natürliches Problem aus NPI bekannt
(Kandidaten: Primfaktorzerlegung, Graphisomorphie)

Und jetzt?

- definiere Hierarchie an Komplexitätsklassen

Bisherige FPT-Reduktionen



Lässt sich die letzte Reduktion auch umkehren?

- wissen wir nicht; Vermutung: nicht umkehrbar
- DOMINATING SET ist also (vermutlich) echt schwerer als CLIQUE oder INDEPENDENT SET
($DS \in \text{FPT} \Rightarrow IS \in \text{FPT}$, aber nicht umgekehrt)
- wir brauchen also einen abgestuften Begriff der Schwere

Vergleich mit der Komplexitätsklasse P

- hier reicht üblicherweise ein Begriff: NP-Schwere
- es gibt aber auch intermediate-Probleme (angenommen $P \neq \text{NP}$)
- es ist allerdings kein natürliches Problem aus NPI bekannt
(Kandidaten: Primfaktorzerlegung, Graphisomorphie)

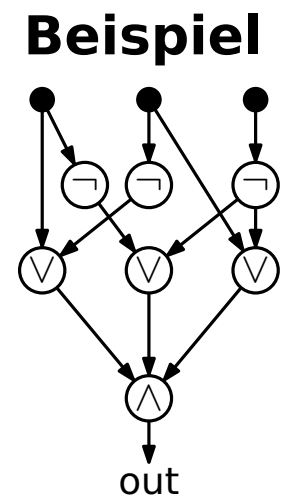
Und jetzt?

- definiere Hierarchie an Komplexitätsklassen
- benutze dazu prototypische vollständige Probleme für jede Stufe

WEIGHTED CIRCUIT SATISFIABILITY

Boolesche Schaltkreise

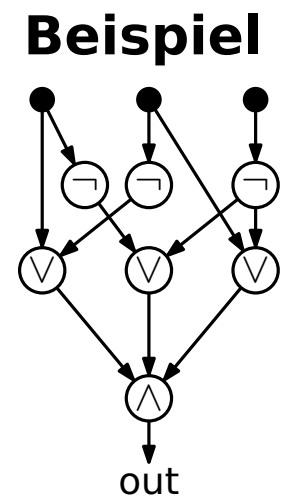
- gerichteter azyklischer Graph (DAG) mit folgenden Knotentypen:



WEIGHTED CIRCUIT SATISFIABILITY



Boolesche Schaltkreise

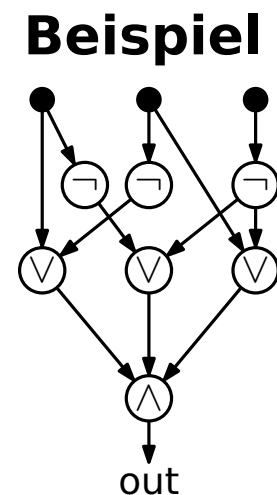
- gerichteter azyklischer Graph (DAG) mit folgenden Knotentypen:
 - Quellen (Eingangsgrad 0) sind **Eingabeknoten** ●



WEIGHTED CIRCUIT SATISFIABILITY

Boolesche Schaltkreise

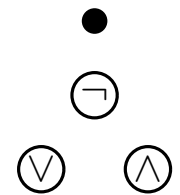
- gerichteter azyklischer Graph (DAG) mit folgenden Knotentypen:
 - Quellen (Eingangsgrad 0) sind **Eingabeknoten** 
 - **Negationsknoten** haben Eingangsgrad 1 



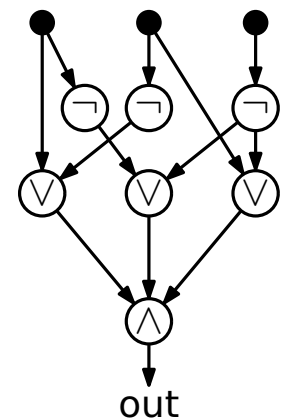
WEIGHTED CIRCUIT SATISFIABILITY

Boolesche Schaltkreise

- gerichteter azyklischer Graph (DAG) mit folgenden Knotentypen:
 - Quellen (Eingangsgrad 0) sind **Eingabeknoten**
 - **Negationsknoten** haben Eingangsgrad 1
 - **und-** bzw. **oder-Knoten** haben Eingangsgrad ≥ 2



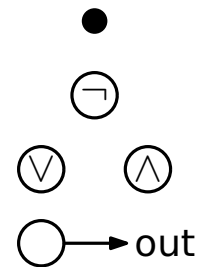
Beispiel



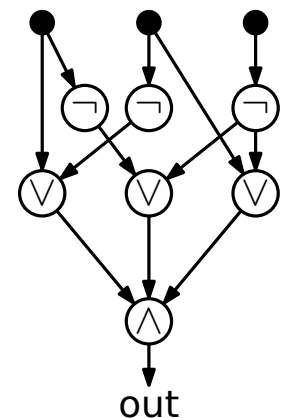
WEIGHTED CIRCUIT SATISFIABILITY

Boolesche Schaltkreise

- gerichteter azyklischer Graph (DAG) mit folgenden Knotentypen:
 - Quellen (Eingangsgrad 0) sind **Eingabeknoten**
 - **Negationsknoten** haben Eingangsgrad 1
 - **und-** bzw. **oder-Knoten** haben Eingangsgrad ≥ 2
 - eine Senke (Ausgangsgrad 0) ist der **Ausgabeknoten**



Beispiel

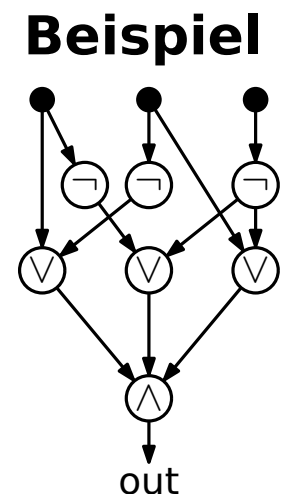


WEIGHTED CIRCUIT SATISFIABILITY

Boolesche Schaltkreise





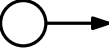
- gerichteter azyklischer Graph (DAG) mit folgenden Knotentypen:
 - Quellen (Eingangsgrad 0) sind **Eingabeknoten** ●
 - **Negationsknoten** haben Eingangsgrad 1 ⊖
 - **und-** bzw. **oder-Knoten** haben Eingangsgrad ≥ 2 ⊓ ⊔
 - eine Senke (Ausgangsgrad 0) ist der **Ausgabeknoten** ○ → out

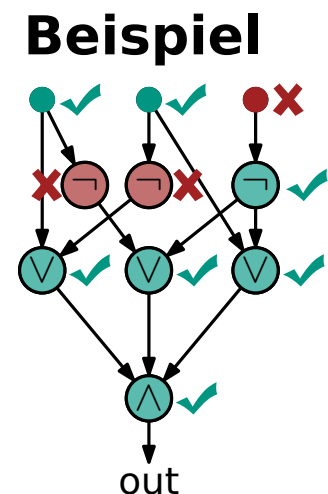
- eine Belegung der Eingabeknoten mit 0 bzw. 1 ergibt Belegung für alle anderen Knoten (auf die offensichtliche Art und Weise)
- eine Belegung ist **erfüllend**, wenn der Ausgabeknoten Wert 1 hat



WEIGHTED CIRCUIT SATISFIABILITY





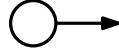
Boolesche Schaltkreise

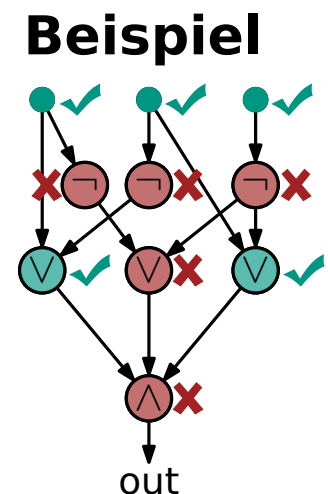
- gerichteter azyklischer Graph (DAG) mit folgenden Knotentypen:
 - Quellen (Eingangsgrad 0) sind **Eingabeknoten** 
 - **Negationsknoten** haben Eingangsgrad 1 
 - **und-** bzw. **oder-Knoten** haben Eingangsgrad ≥ 2  
 - eine Senke (Ausgangsgrad 0) ist der **Ausgabeknoten**  out
- eine Belegung der Eingabeknoten mit 0 bzw. 1 ergibt Belegung für alle anderen Knoten (auf die offensichtliche Art und Weise)
- eine Belegung ist **erfüllend**, wenn der Ausgabeknoten Wert 1 hat



WEIGHTED CIRCUIT SATISFIABILITY





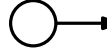
Boolesche Schaltkreise

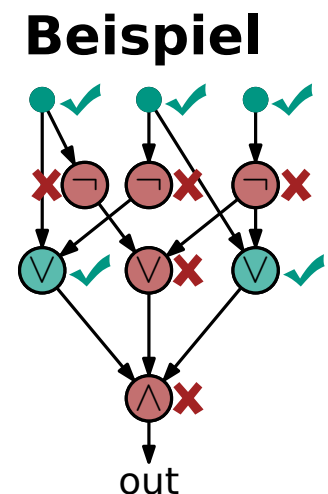
- gerichteter azyklischer Graph (DAG) mit folgenden Knotentypen:
 - Quellen (Eingangsgrad 0) sind **Eingabeknoten** 
 - **Negationsknoten** haben Eingangsgrad 1 
 - **und-** bzw. **oder-Knoten** haben Eingangsgrad ≥ 2  
 - eine Senke (Ausgangsgrad 0) ist der **Ausgabeknoten**  out
- eine Belegung der Eingabeknoten mit 0 bzw. 1 ergibt Belegung für alle anderen Knoten (auf die offensichtliche Art und Weise)
- eine Belegung ist **erfüllend**, wenn der Ausgabeknoten Wert 1 hat



WEIGHTED CIRCUIT SATISFIABILITY





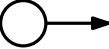
Boolesche Schaltkreise

- gerichteter azyklischer Graph (DAG) mit folgenden Knotentypen:
 - Quellen (Eingangsgrad 0) sind **Eingabeknoten** 
 - **Negationsknoten** haben Eingangsgrad 1 
 - **und-** bzw. **oder-Knoten** haben Eingangsgrad ≥ 2  
 - eine Senke (Ausgangsgrad 0) ist der **Ausgabeknoten**  out
- eine Belegung der Eingabeknoten mit 0 bzw. 1 ergibt Belegung für alle anderen Knoten (auf die offensichtliche Art und Weise)
- eine Belegung ist **erfüllend**, wenn der Ausgabeknoten Wert 1 hat
- **Gewicht** einer Belegung = Anzahl verwendeter 1en



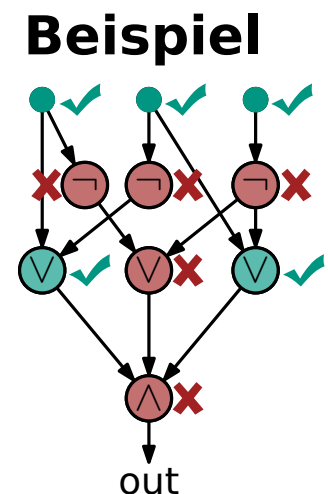
WEIGHTED CIRCUIT SATISFIABILITY

Boolesche Schaltkreise

- gerichteter azyklischer Graph (DAG) mit folgenden Knotentypen:
 - Quellen (Eingangsgrad 0) sind **Eingabeknoten** 
 - **Negationsknoten** haben Eingangsgrad 1 
 - **und-** bzw. **oder-Knoten** haben Eingangsgrad ≥ 2  
 - eine Senke (Ausgangsgrad 0) ist der **Ausgabeknoten**  out
- eine Belegung der Eingabeknoten mit 0 bzw. 1 ergibt Belegung für alle anderen Knoten (auf die offensichtliche Art und Weise)
- eine Belegung ist **erfüllend**, wenn der Ausgabeknoten Wert 1 hat
- **Gewicht** einer Belegung = Anzahl verwendeter 1en

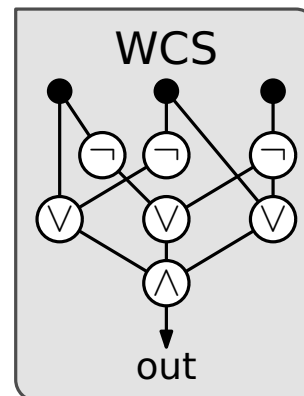
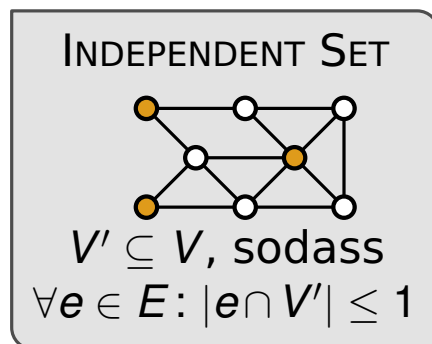
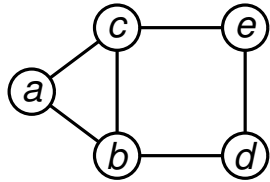
Problem: WEIGHTED CIRCUIT SATISFIABILITY (WCS)

Gegeben ein Boolescher Schaltkreis und ein Parameter k .
Gibt es eine erfüllende Belegung mit Gewicht genau k ?



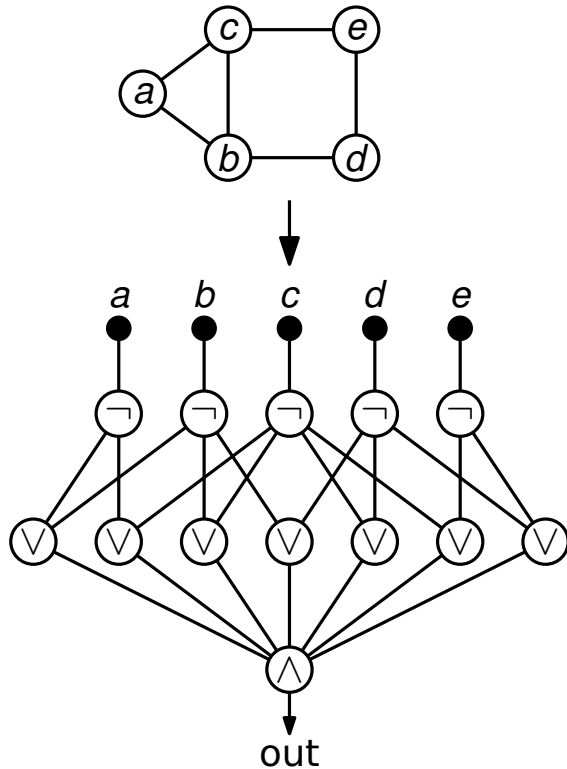
Reduktionen so weit das Auge reicht

INDEPENDENT SET \rightarrow WCS

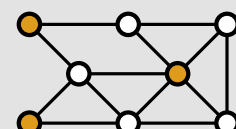


Reduktionen so weit das Auge reicht

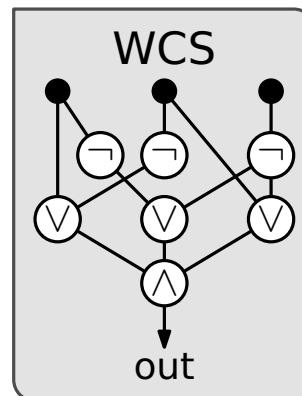
INDEPENDENT SET \rightarrow WCS



INDEPENDENT SET

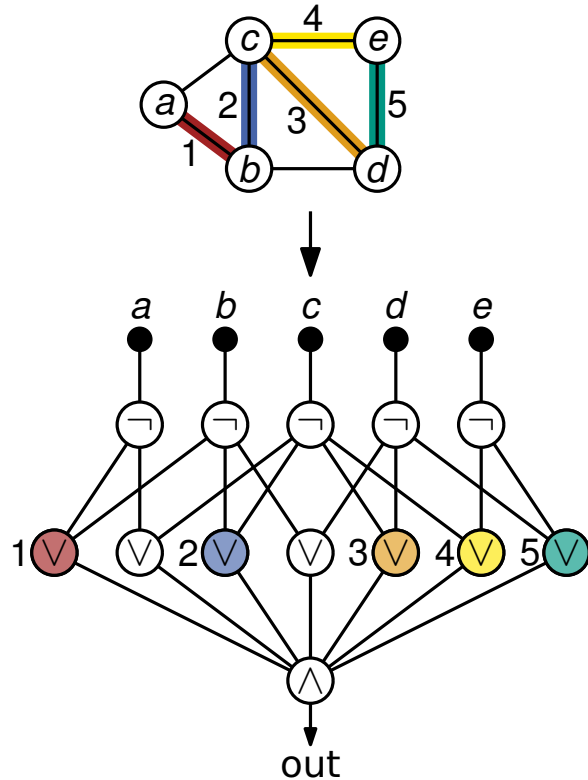


$V' \subseteq V$, sodass
 $\forall e \in E: |e \cap V'| \leq 1$

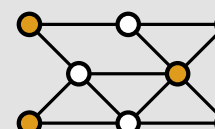


Reduktionen so weit das Auge reicht

INDEPENDENT SET \rightarrow WCS

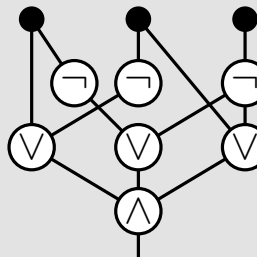


INDEPENDENT SET



$V' \subseteq V$, sodass
 $\forall e \in E: |e \cap V'| \leq 1$

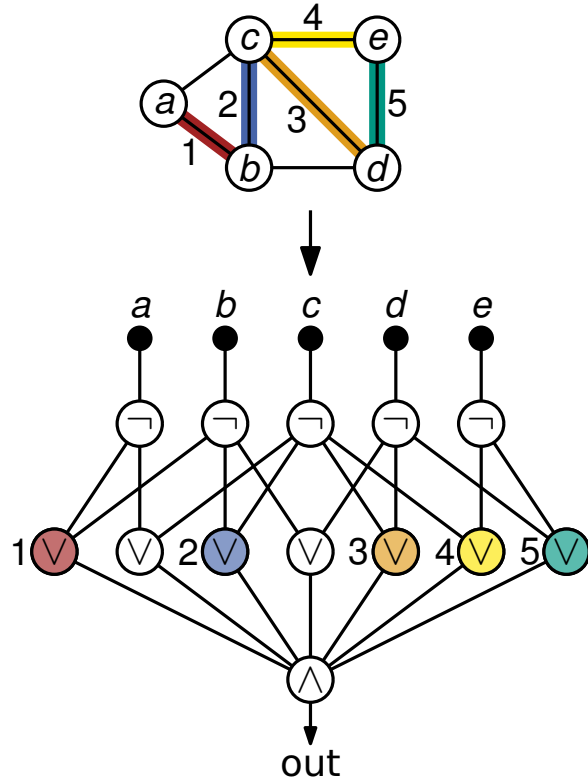
WCS



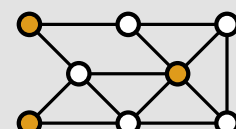
out

Reduktionen so weit das Auge reicht

INDEPENDENT SET \rightarrow WCS

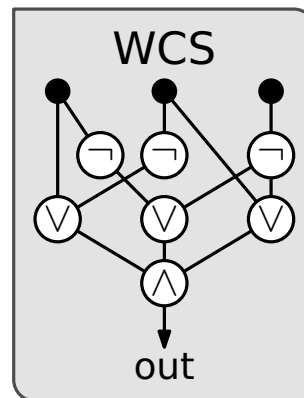
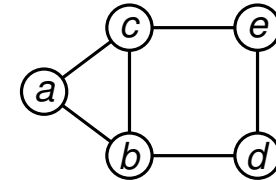


INDEPENDENT SET

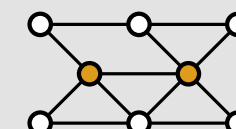


$V' \subseteq V$, sodass
 $\forall e \in E: |e \cap V'| \leq 1$

DOMINATING SET \rightarrow WCS



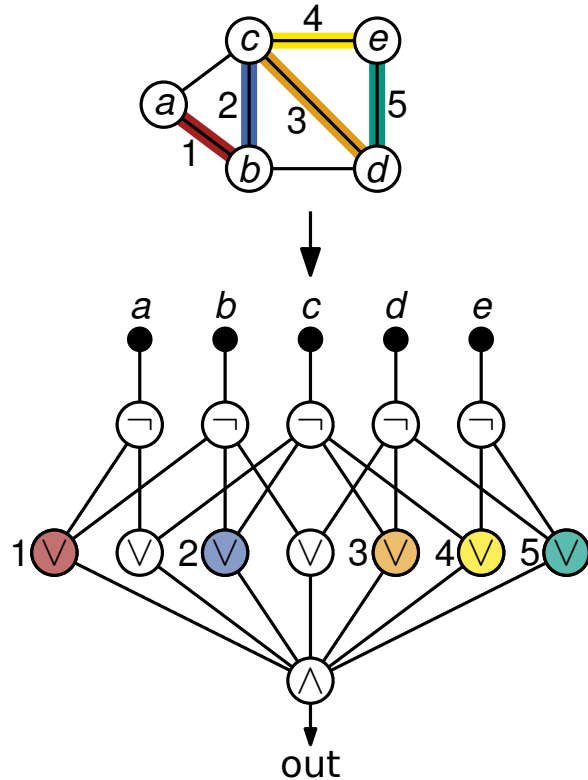
DOMINATING SET



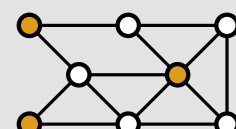
$V' \subseteq V$, sodass
 $\forall v \in V: |N[v] \cap V'| \geq 1$

Reduktionen so weit das Auge reicht

INDEPENDENT SET \rightarrow WCS

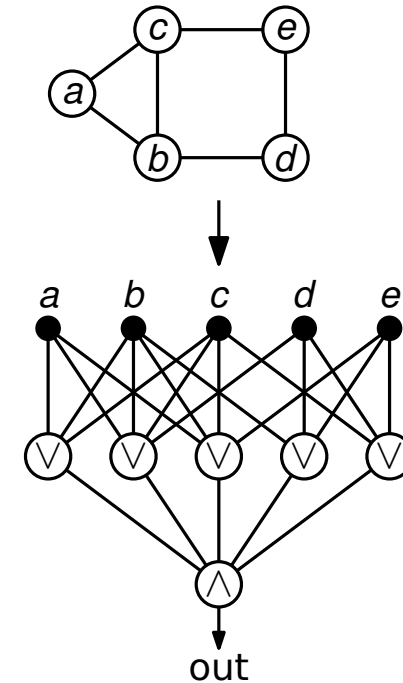


INDEPENDENT SET

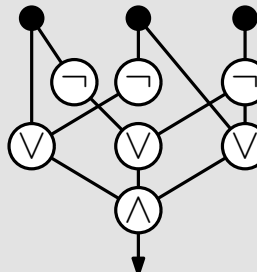


$V' \subseteq V$, sodass
 $\forall e \in E: |e \cap V'| \leq 1$

DOMINATING SET \rightarrow WCS

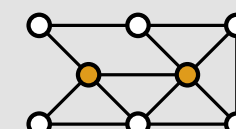


WCS



out

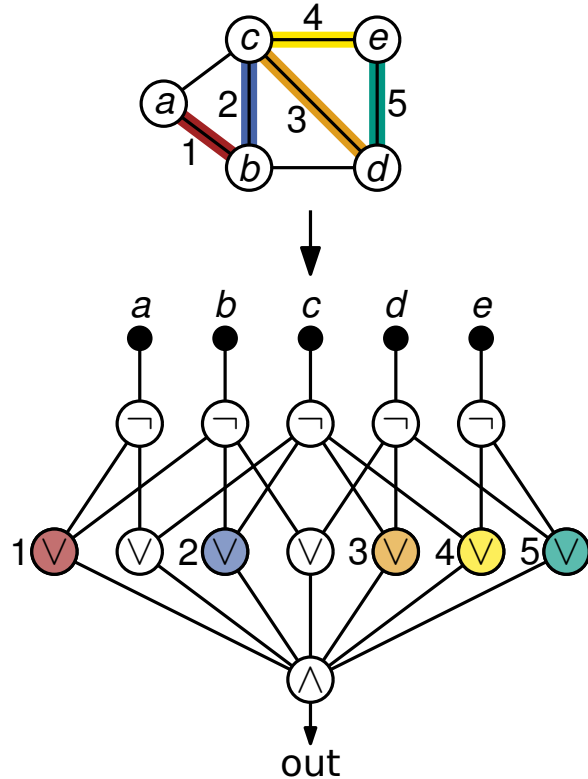
DOMINATING SET



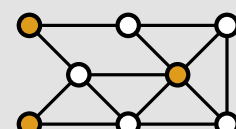
$V' \subseteq V$, sodass
 $\forall v \in V: |N[v] \cap V'| \geq 1$

Reduktionen so weit das Auge reicht

INDEPENDENT SET \rightarrow WCS

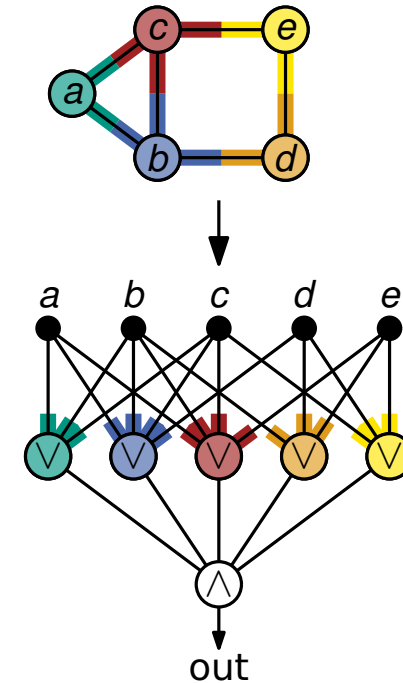


INDEPENDENT SET

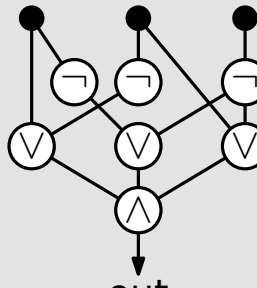


$V' \subseteq V$, sodass
 $\forall e \in E: |e \cap V'| \leq 1$

DOMINATING SET \rightarrow WCS

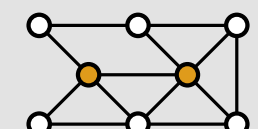


WCS



out

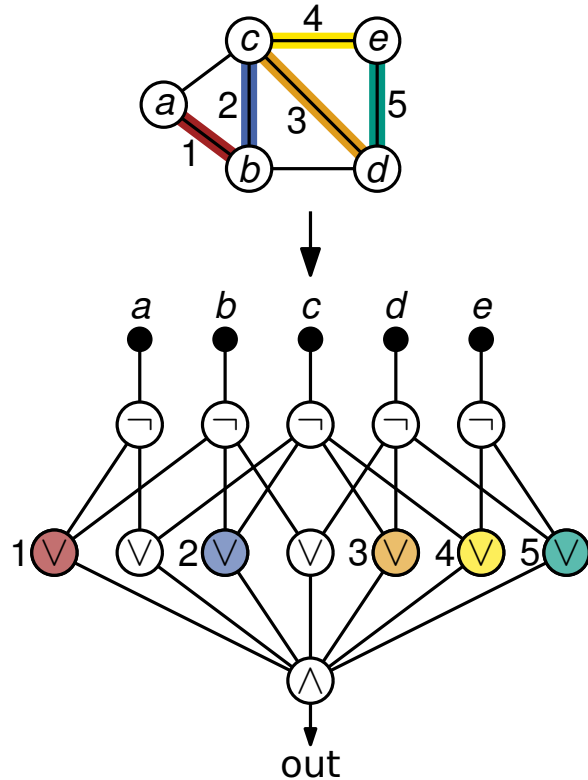
DOMINATING SET



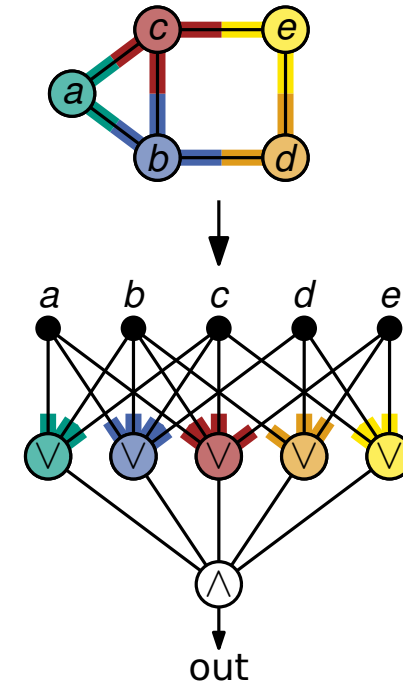
$V' \subseteq V$, sodass
 $\forall v \in V: |N[v] \cap V'| \geq 1$

Reduktionen so weit das Auge reicht

INDEPENDENT SET \rightarrow WCS



DOMINATING SET \rightarrow WCS

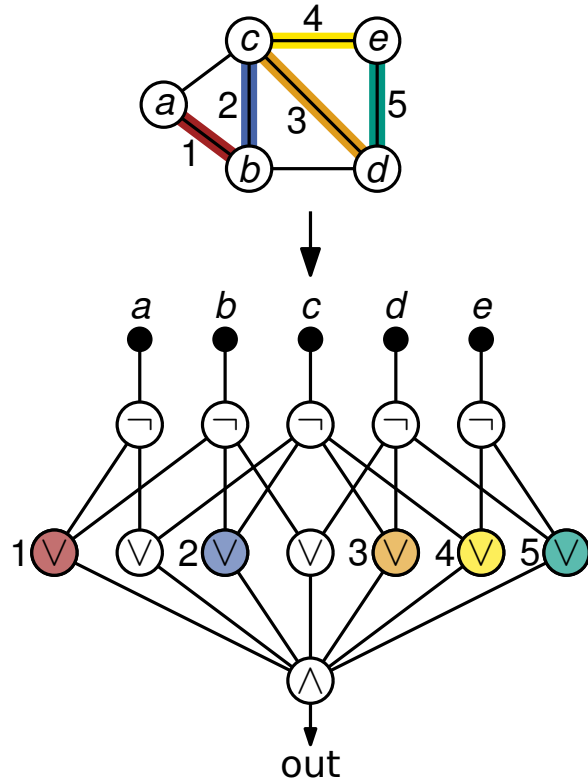


Beobachtungen

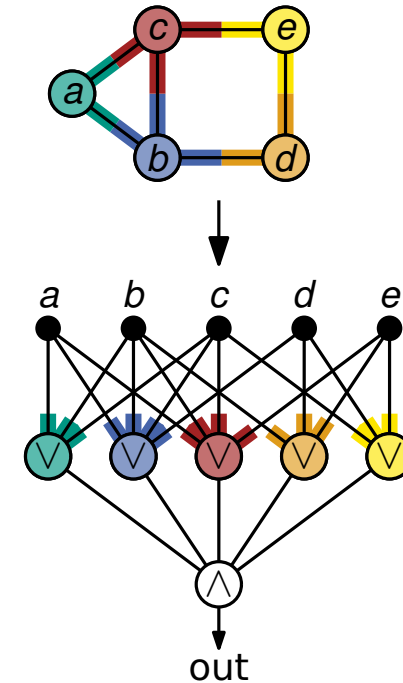
- die Schaltkreise haben beide konstante Tiefe

Reduktionen so weit das Auge reicht

INDEPENDENT SET \rightarrow WCS



DOMINATING SET \rightarrow WCS

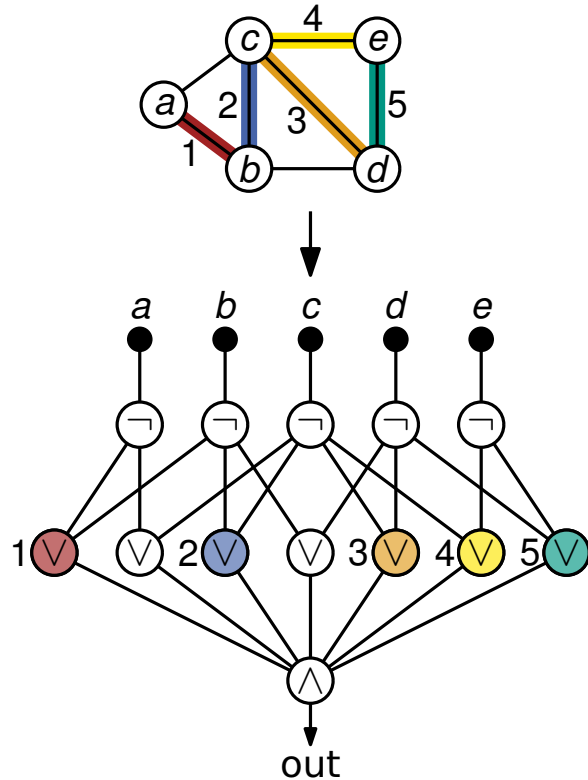


Beobachtungen

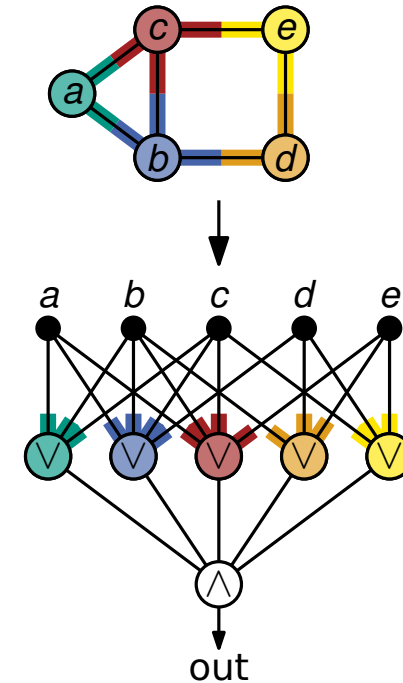
- die Schaltkreise haben beide konstante Tiefe
- der Schaltkreis für DS enthält mehr Knoten mit Eingangsgrad > 2

Reduktionen so weit das Auge reicht

INDEPENDENT SET \rightarrow WCS



DOMINATING SET \rightarrow WCS

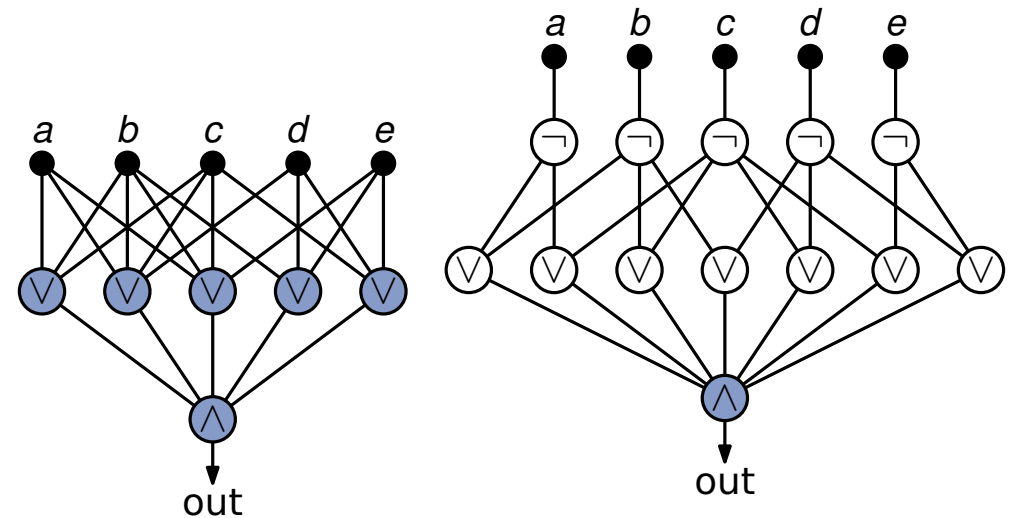
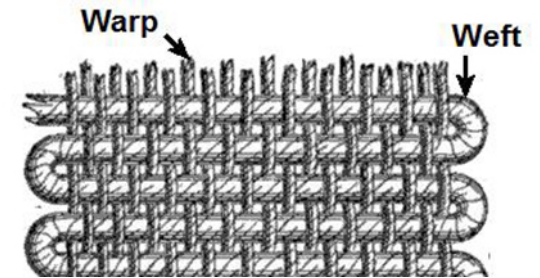


Beobachtungen

- die Schaltkreise haben beide konstante Tiefe
- der Schaltkreis für DS enthält mehr Knoten mit Eingangsgrad > 2
- ist DS deshalb schwerer (bezüglich FPT) als IS?

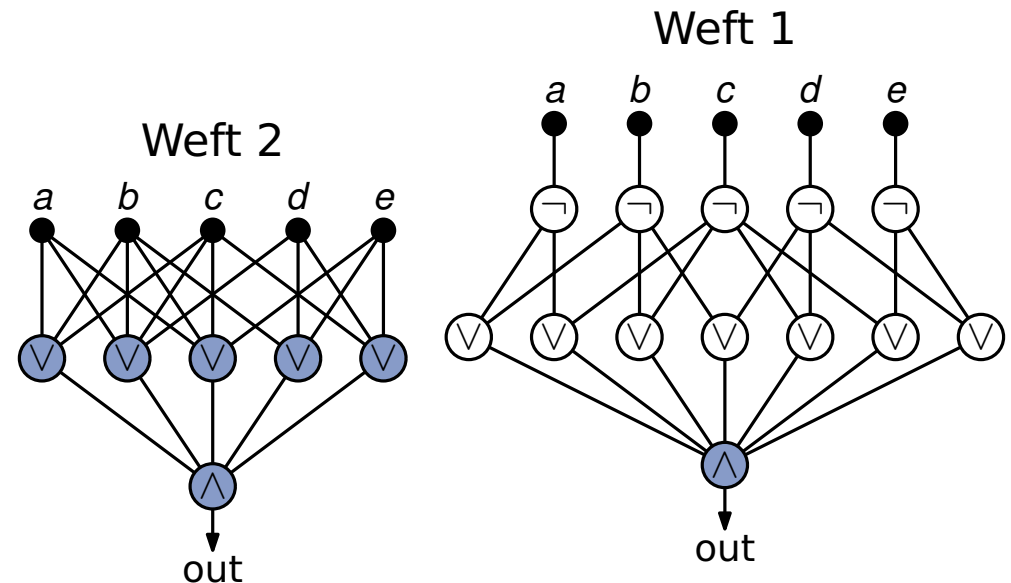
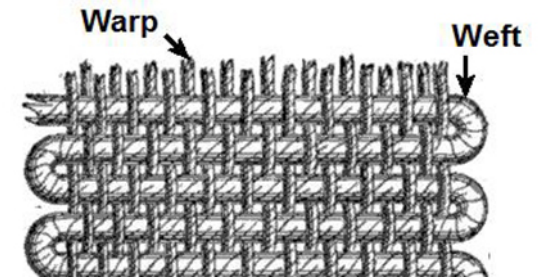
Definition

Der **Weft** eines Booleschen Schaltkreises ist die maximale Anzahl an Knoten mit Eingangsgrad > 2 auf einem gerichteten Pfad.



Definition

Der **Weft** eines Booleschen Schaltkreises ist die maximale Anzahl an Knoten mit Eingangsgrad > 2 auf einem gerichteten Pfad.

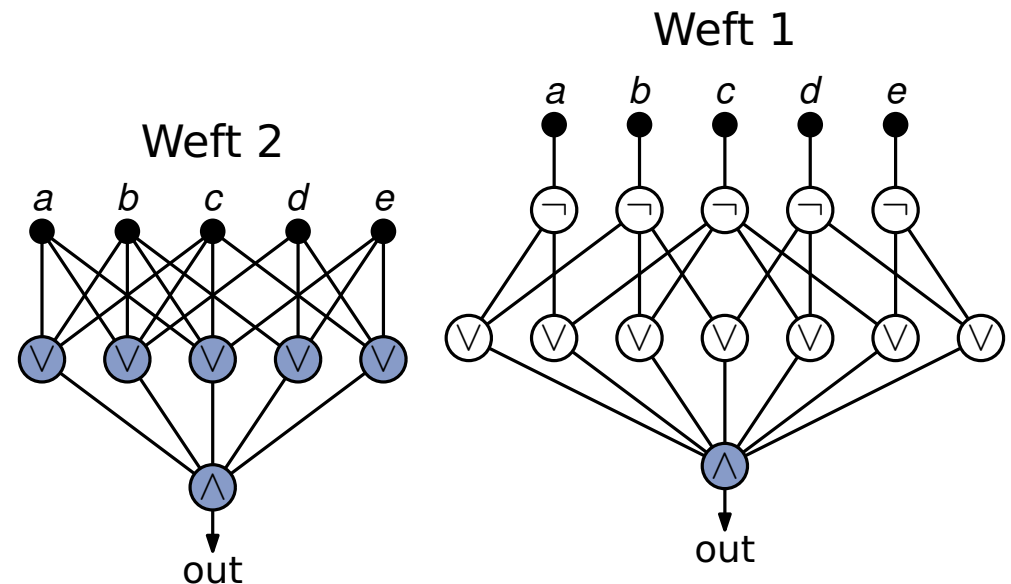
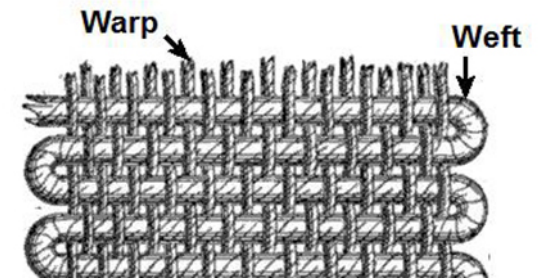


Definition

Der **Weft** eines Booleschen Schaltkreises ist die maximale Anzahl an Knoten mit Eingangsgrad > 2 auf einem gerichteten Pfad.

Problem

WCS $[t]$ ist WCS eingeschränkt auf Schaltkreise mit konstanter Tiefe und Weft maximal t .



Weft

Definition

Der **Weft** eines Booleschen Schaltkreises ist die maximale Anzahl an Knoten mit Eingangsgrad > 2 auf einem gerichteten Pfad.

Problem

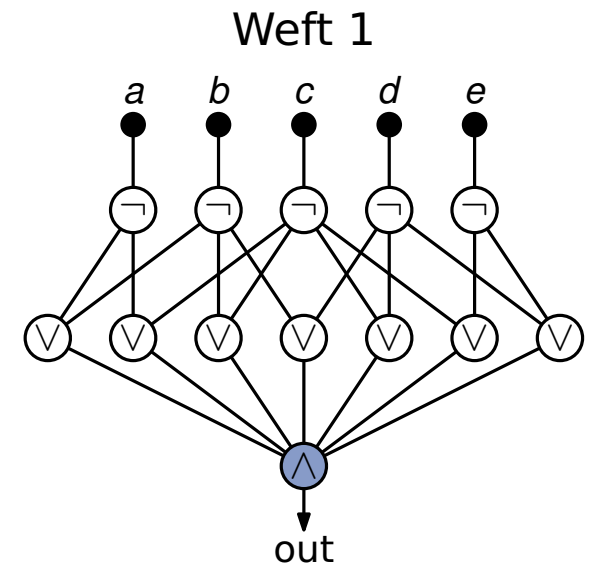
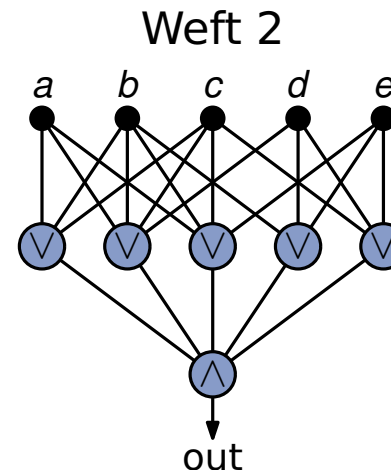
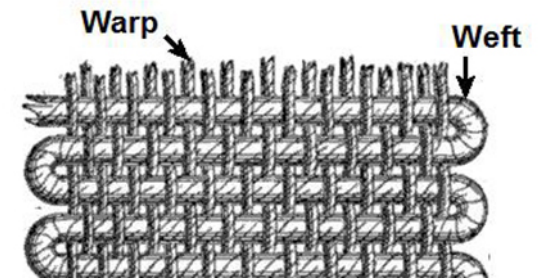
WCS $[t]$ ist WCS eingeschränkt auf Schaltkreise mit konstanter Tiefe und Weft maximal t .

Definition

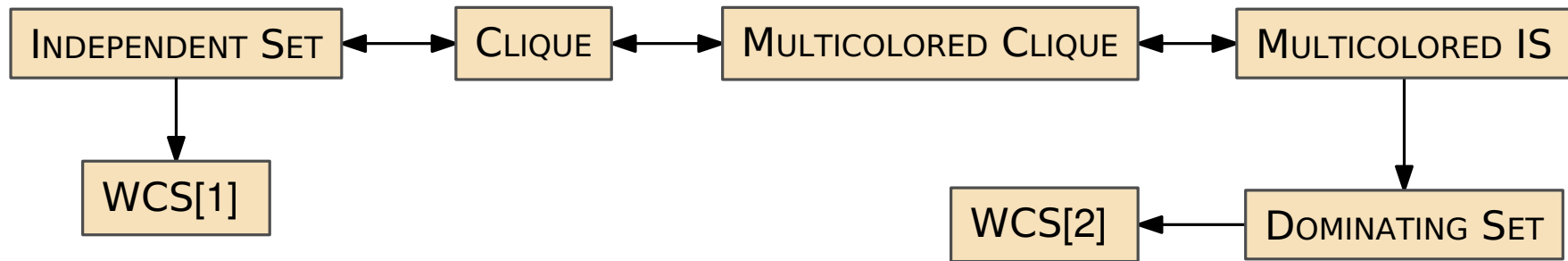
Die Klasse **W** $[t]$ enthält die Probleme, die eine parametrisierte Reduktion auf WCS $[t]$ zulassen.

Eben gesehen

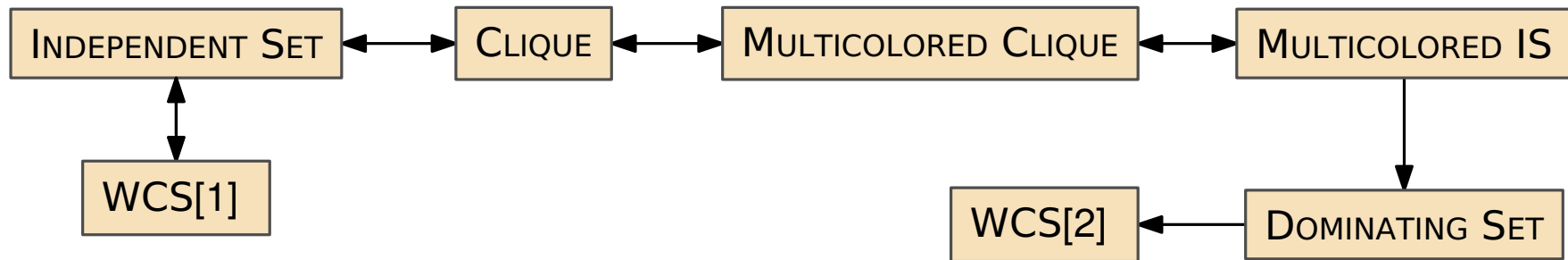
- INDEPENDENT SET $\in W[1]$
- DOMINATING SET $\in W[2]$



Bisherige FPT-Reduktionen



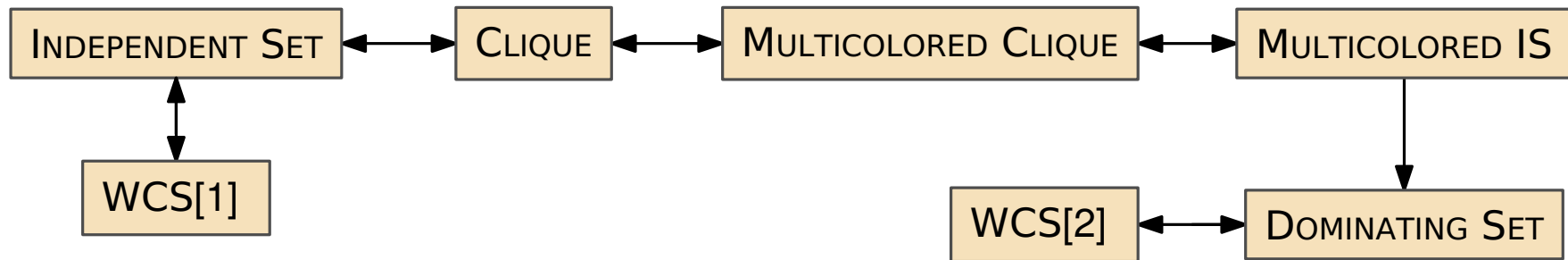
Bisherige FPT-Reduktionen



Weitere Reduktionen

- man kann auch WCS[1] auf INDEPENDENT SET reduzieren

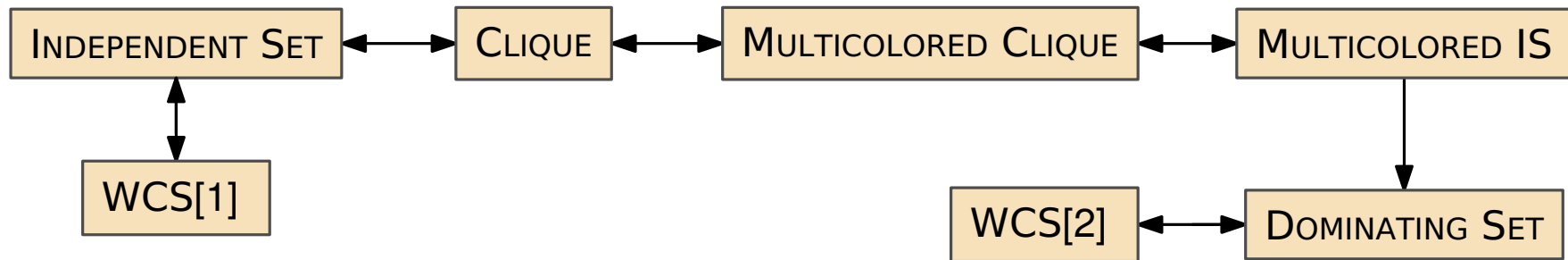
Bisherige FPT-Reduktionen



Weitere Reduktionen

- man kann auch WCS[1] auf INDEPENDENT SET reduzieren
- und WCS[2] auf DOMINATING SET

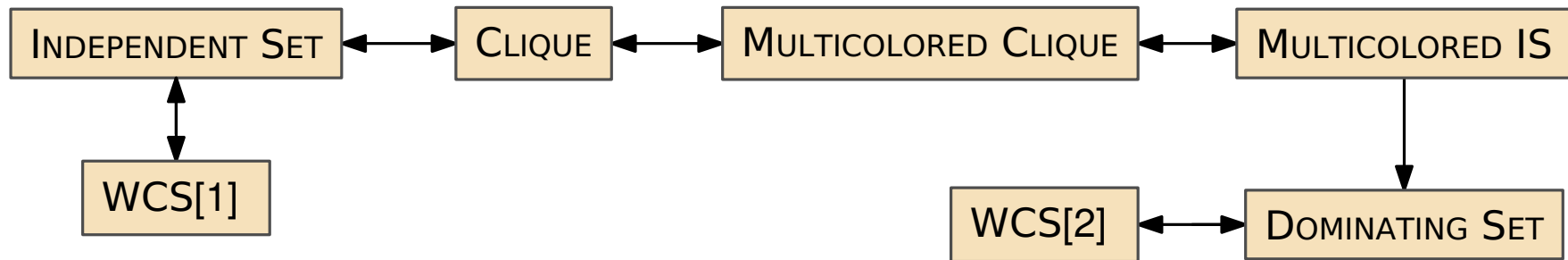
Bisherige FPT-Reduktionen



Weitere Reduktionen

- man kann auch $WCS[1]$ auf INDEPENDENT SET reduzieren
- und $WCS[2]$ auf DOMINATING SET
- man kann also jedes Problem aus $W[1]$ auf IS reduzieren
- solche Probleme nennt man $W[1]$ -vollständig

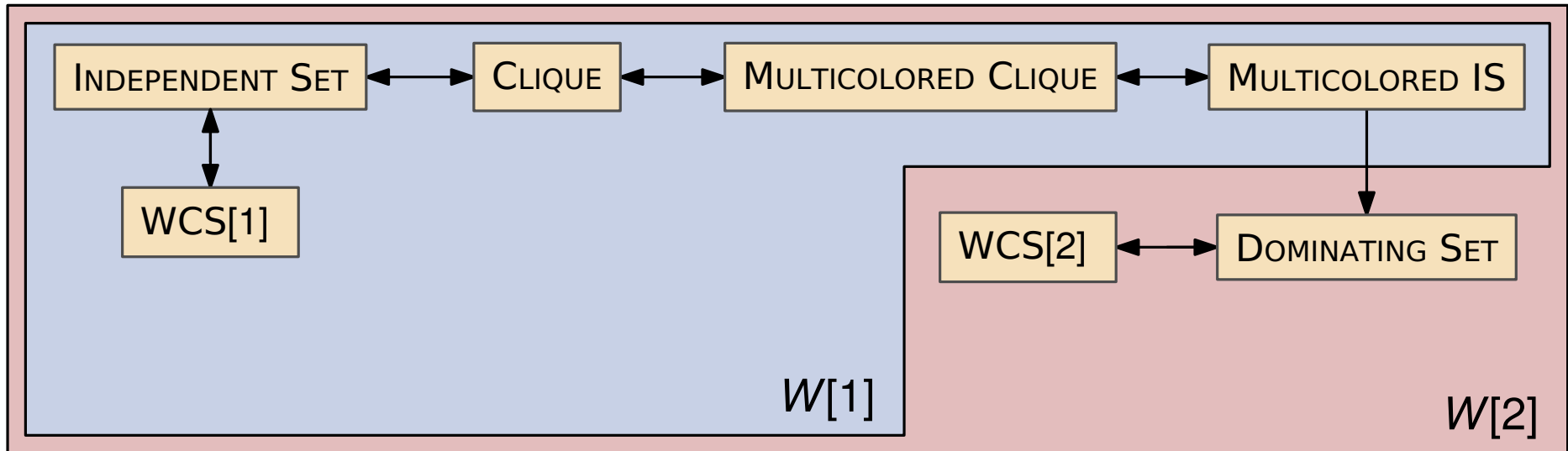
Bisherige FPT-Reduktionen



Weitere Reduktionen

- man kann auch WCS[1] auf INDEPENDENT SET reduzieren
- und WCS[2] auf DOMINATING SET
- man kann also jedes Problem aus $W[1]$ auf IS reduzieren
- solche Probleme nennt man $W[1]$ -vollständig
- analog ist DS $W[2]$ -vollständig

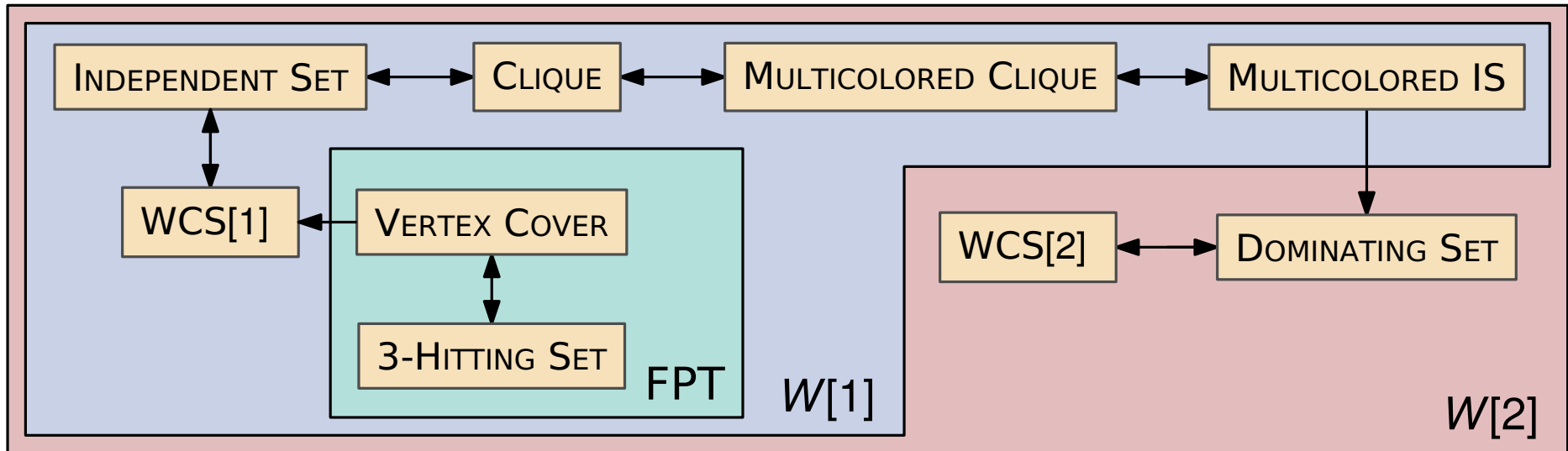
Bisherige FPT-Reduktionen



Weitere Reduktionen

- man kann auch WCS[1] auf INDEPENDENT SET reduzieren
- und WCS[2] auf DOMINATING SET
- man kann also jedes Problem aus $W[1]$ auf IS reduzieren
- solche Probleme nennt man $W[1]$ -vollständig
- analog ist DS $W[2]$ -vollständig
- beachte: $W[1] \subseteq W[2]$ Warum?

Bisherige FPT-Reduktionen



Weitere Reduktionen

- man kann auch WCS[1] auf INDEPENDENT SET reduzieren
- und WCS[2] auf DOMINATING SET
- man kann also jedes Problem aus $W[1]$ auf IS reduzieren
- solche Probleme nennt man $W[1]$ -vollständig
- analog ist DS $W[2]$ -vollständig
- beachte: $W[1] \subseteq W[2]$ Warum?
- außerdem gilt: $FPT \subseteq W[1] \subseteq W[2]$ Warum?

Zusammenfassung

Die W-Hierarchie

- Komplexitätsklassen $FPT \subseteq W[1] \subseteq W[2] \subseteq W[3] \subseteq \dots$
- $W[t]$ definiert über ein prototypisches vollständiges Problem $WCS[t]$:
 $\mathcal{L} \in W[t] \Leftrightarrow \mathcal{L}$ auf $WCS[t]$ reduzierbar (mittels FPT-Reduktion)

Zusammenfassung

Die W-Hierarchie

- Komplexitätsklassen $FPT \subseteq W[1] \subseteq W[2] \subseteq W[3] \subseteq \dots$
- $W[t]$ definiert über ein prototypisches vollständiges Problem $WCS[t]$:
 $\mathcal{L} \in W[t] \Leftrightarrow \mathcal{L}$ auf $WCS[t]$ reduzierbar (mittels FPT-Reduktion)
- Vermutung: jede der Inklusionen ist echt

Zusammenfassung

Die W -Hierarchie

- Komplexitätsklassen $FPT \subseteq W[1] \subseteq W[2] \subseteq W[3] \subseteq \dots$
- $W[t]$ definiert über ein prototypisches vollständiges Problem $WCS[t]$:
 $\mathcal{L} \in W[t] \Leftrightarrow \mathcal{L}$ auf $WCS[t]$ reduzierbar (mittels FPT-Reduktion)
- Vermutung: jede der Inklusionen ist echt

Ist mein $W[t]$ -Vollständigkeitsbeweis nutzlos, falls $W[t] = FPT$?

Zusammenfassung

Die W -Hierarchie

- Komplexitätsklassen $FPT \subseteq W[1] \subseteq W[2] \subseteq W[3] \subseteq \dots$
- $W[t]$ definiert über ein prototypisches vollständiges Problem $WCS[t]$:
 $\mathcal{L} \in W[t] \Leftrightarrow \mathcal{L}$ auf $WCS[t]$ reduzierbar (mittels FPT-Reduktion)
- Vermutung: jede der Inklusionen ist echt

Ist mein $W[t]$ -Vollständigkeitsbeweis nutzlos, falls $W[t] = FPT$?

- man weiß trotzdem: ich finde keinen FPT-Algo, weil sich da ein fundamentales ungelöstes Problem versteckt, nicht weil ich zu dumm bin

Zusammenfassung

Die W-Hierarchie

- Komplexitätsklassen $FPT \subseteq W[1] \subseteq W[2] \subseteq W[3] \subseteq \dots$
- $W[t]$ definiert über ein prototypisches vollständiges Problem $WCS[t]$:
 $\mathcal{L} \in W[t] \Leftrightarrow \mathcal{L}$ auf $WCS[t]$ reduzierbar (mittels FPT-Reduktion)
- Vermutung: jede der Inklusionen ist echt

Ist mein $W[t]$ -Vollständigkeitsbeweis nutzlos, falls $W[t] = FPT$?

- man weiß trotzdem: ich finde keinen FPT-Algo, weil sich da ein fundamentales ungelöstes Problem versteckt, nicht weil ich zu dumm bin
- findet man einen FPT-Algo für ein vollständiges Problem, so liefern die Reduktionen weitere FPT-Algos

Zusammenfassung

Die W -Hierarchie

- Komplexitätsklassen $FPT \subseteq W[1] \subseteq W[2] \subseteq W[3] \subseteq \dots$
- $W[t]$ definiert über ein prototypisches vollständiges Problem $WCS[t]$:
 $\mathcal{L} \in W[t] \Leftrightarrow \mathcal{L}$ auf $WCS[t]$ reduzierbar (mittels FPT-Reduktion)
- Vermutung: jede der Inklusionen ist echt

Ist mein $W[t]$ -Vollständigkeitsbeweis nutzlos, falls $W[t] = FPT$?

- man weiß trotzdem: ich finde keinen FPT-Algo, weil sich da ein fundamentales ungelöstes Problem versteckt, nicht weil ich zu dumm bin
- findet man einen FPT-Algo für ein vollständiges Problem, so liefern die Reduktionen weitere FPT-Algos

Wie zeige ich, dass mein Problem schwer ist?

- reduziere von anderen schweren Problemen

Zusammenfassung

Die W-Hierarchie

- Komplexitätsklassen $FPT \subseteq W[1] \subseteq W[2] \subseteq W[3] \subseteq \dots$
- $W[t]$ definiert über ein prototypisches vollständiges Problem $WCS[t]$:
 $\mathcal{L} \in W[t] \Leftrightarrow \mathcal{L}$ auf $WCS[t]$ reduzierbar (mittels FPT-Reduktion)
- Vermutung: jede der Inklusionen ist echt

Ist mein $W[t]$ -Vollständigkeitsbeweis nutzlos, falls $W[t] = FPT$?

- man weiß trotzdem: ich finde keinen FPT-Algo, weil sich da ein fundamentales ungelöstes Problem versteckt, nicht weil ich zu dumm bin
- findet man einen FPT-Algo für ein vollständiges Problem, so liefern die Reduktionen weitere FPT-Algos

Wie zeige ich, dass mein Problem schwer ist?

- reduziere von anderen schweren Problemen
- Reduktion von INDEPENDENT SET oder CLIQUE zeigt $W[1]$ -Schwere

Zusammenfassung

Die W -Hierarchie

- Komplexitätsklassen $FPT \subseteq W[1] \subseteq W[2] \subseteq W[3] \subseteq \dots$
- $W[t]$ definiert über ein prototypisches vollständiges Problem $WCS[t]$:
 $\mathcal{L} \in W[t] \Leftrightarrow \mathcal{L}$ auf $WCS[t]$ reduzierbar (mittels FPT-Reduktion)
- Vermutung: jede der Inklusionen ist echt

Ist mein $W[t]$ -Vollständigkeitsbeweis nutzlos, falls $W[t] = FPT$?

- man weiß trotzdem: ich finde keinen FPT-Algo, weil sich da ein fundamentales ungelöstes Problem versteckt, nicht weil ich zu dumm bin
- findet man einen FPT-Algo für ein vollständiges Problem, so liefern die Reduktionen weitere FPT-Algos

Wie zeige ich, dass mein Problem schwer ist?

- reduziere von anderen schweren Problemen
- Reduktion von INDEPENDENT SET oder CLIQUE zeigt $W[1]$ -Schwere
- Reduktion von DOMINATING SET oder HITTING SET zeigt $W[2]$ -Schwere